An ontology-based approach for integrating engineering workflows for industrial assembly automation systems

Mussawar Ahmad

Thesis submitted to the University of Warwick in partial fulfilment for the degree of Doctor of Philosophy

August 2017

Abstract

Modern manufacturing organisations face a number of external challenges as the customer-base is more varied, more knowledgeable, and has a broader range of requirements. This has given rise to paradigms such as mass customisation and product personalisation. Internally, businesses must manage multidisciplinary teams that must work together to achieve a common goal despite spanning multiple domains, organisations, and due to improved communication technologies, countries.

The motivation for this research is to therefore understand firstly how the multiplicity of stakeholders come together to realise the ever increasing and ever more complex number of product variants that manufacturing systems must now realise. The lack of integration of engineering tools and methods is identified to be one of the barriers to smooth engineering workflows and thus one of the key challenges faced in the current dynamic market.

To address this problem, this research builds upon previous works that propose domain ontologies for representing knowledge in a way that is both machine and human readable, facilitating interoperability between engineering software. In addition to this, the research develops a novel Skill model that brings the domain ontologies into a practical, implementable framework that complements existing industrial workflows. The focus of this thesis is the domain of industrial assembly automation systems due to the role this stage of manufacturing plays in realising product variety. Therefore, the proposed ontological models and framework are applied to product assembly scenarios.

The key contributions of this work are the consolidation of domain ontologies with a Skill model within the context of assembly systems engineering, development of a broader framework for the ontologies to sit within that complements existing workflows. In addition, the research demonstrates how the framework can be applied to connect assembly process planning activities with machine control logic to identify and rectify inconsistencies as new products are introduced.

In summary, the thesis identifies the shortcomings of existing ontological models within the context of manufacturing, develops new models to address those shortcoming, and develops new, useful ways for ontological models to be used to address industrial problems by integrating them with virtual engineering tools.

List of Publications

Over the course of the PhD research project the author has published a number of works, with some still in progress.

First Author

- Ahmad, M., Zhang, J., Ahmad, B., Harrison, R., "*Connecting assembly process planning with machine control software using virtual engineering tools and semantic web technologies"*, Submitted to the Journal of Robotics and Computer Integrated Manufacturing (July 2017)
- Ahmad, M., Ferrer, B. R., Ahmad, B., Vera, D., Martinez Lastra, J. L., Harrison, R., "*Knowledge-Based PPR Modelling for Assembly Automation",* Submitted to the CIRP Journal of Manufacturing Science and Technology (accepted – January 2018)
- Ahmad, M., Ferrer, B. R., Ahmad, B., Martinez Lastra, J. L., Harrison, R., (2017), *"Ensuring the consistency between assembly process planning and machine control software"*, Proceedings of the 15th International Conference on Industrial Informatics (INDIN 2017), 24-26th July 2017, Emden, Germany
- Ahmad, M., Harrison, R., Meredith, J., Bindel, A., Todd, B., (2017), "*Validation of a fuel cell compression spring equivalent model using polarisation data*", International Journal of Hydrogen Energy
- Ahmad, M., Ahmad, B., Harrison, R., Alkan, B., Vera, D., Meredith, J., Bindel, A., (2016), *"A framework for automatically realizing assembly sequence changes in a virtual manufacturing environment"* Proceedings of the 26th CIRP Design conference, 15-17th June 2016, Stockholm, Sweden
- Ahmad, M., Ahmad, B., Alkan, B., Vera, D., Harrison, R., Meredith, J., Bindel, A., (2016), *"Hydrogen fuel cell pick and place assembly systems: Heuristic evaluation of reconfigurability and suitability"*, Proceedings of the 49th CIRP Conference on Manufacturing Systems (CIRP-CMS 2016), 25-27th May 2016, Stuttgart, Germany
- Ahmad, M., Alkan, B., Ahmad, B., Vera, D., Harrison, R., Meredith, J., Bindel, A., (2016), *"The use of a complexity model to facilitate in the selection of a fuel cell assembly sequence"*, Proceedings of the 6th CIRP Conference on Assembly Technologies and Systems (CATS), 16-18th May 2016, Gothenburg, Sweden. **BEST PAPER AWARD**
- Ahmad, M., Harrison, R., Ferrer, B. R., Martinez Lastra, J. L., Meredith, J., Bindel, A., (2015), *"A knowledge-based approach for the selection of assembly equipment*

based on fuel cell component characteristics", Proceedings of the 41st annual conference of the IEEE Industrial Electronics Society (IES) (IECON), 9-12th November 2015, Yokohama, Japan

- Ahmad, M., Harrison, R., Meredith, J., Bindel, A., Todd, B., (2015), *"Analysis of the compression characteristics of a PEM stack, development of an equivalent spring model and recommendations for compression process improvements"*, Proceedings of the 6th International Renewable Energy Congress (IREC), 24-26th March 2015, Sousse, Tunisia

Co-Author

- Alkan, B., Ahmad, M., Vera, D., Harrison, R., *"Complexity in manufacturing systems and its measures: A literature review",* Submitted to the European Journal of Industrial Engineering (accepted – November 2017)
- Alkan, B., Ahmad, M., Vera, D., Harrison, R., *"Heuristic design evaluation of reconfigurable manufacturing systems based on time independent complexity criteria",* Submitted to Journal of Engineering Design (July 2017)
- Ferrer, B. R., Mohammed, W. M., Martinez Lastra, J. L., Ahmad, M., Zhang, J., Harrison, R., Iarovyi, S., *"Comparing Ontologies and Databases: a critical review for lifecycle engineering models in manufacturing"* – in progress
- Afolaranmi, S. O., Ferrer, B. R., Mohammed, W. M., Martinez Lastra, J. L., Ahmad, M., Harrison, R., *"Providing an Access Control layer to Web-Based Applications for the industrial domain; FASTory simulator case"* Proceedings of the 15th International Conference on Industrial Informatics (INDIN 2017), 24-26th July 2017, Emden, Germany
- Konstantinov, S., Ahmad, M., Ananthanarayan, K., Harrison, R., (2017)*"The Cyber-Physical e-machine Manufacturing System: Virtual Engineering for Complete Lifecycle Support"* Proceedings of the 50th CIRP Conference on Manufacturing Systems (CIRP-CMS), 3-5th May 2017, Taichung City, Taiwan. **BEST PAPER AWARD**
- Chinnathai, M. K., Gunther, T., Ahmad, M., Stocker, C., Richter, L., Schreiner, D., Vera, D., Reinhart, G., Harrison, R., (2017) *"An application of physical flexibility and software reconfigurability for the automation of battery module assembly"* Proceedings of the 50th CIRP Conference on Manufacturing Systems (CIRP-CMS), 3-5th May 2017, Taichung City, Taiwan.

- Alkan, B., Vera, D., Ahmad, M., Ahmad, B., Harrison, R., (2016) *"Design evaluation of automated manufacturing processes based on complexity of control logic"* Proceedings of the 26th CIRP Design conference, 15-17th June 2016, Stockholm, Sweden
- Alkan, B., Vera, D., Ahmad, M., Ahmad, B., Harrison, R., (2016) *"A model for complexity assessment in manual assembly operations through predetermined motion time systems",* Proceedings of the 6th CIRP Conference on Assembly Technologies and Systems (CATS), 16-18th May 2016, Gothenburg, Sweden
- Alkan, B., Vera, D., Ahmad, M., Ahmad, B., Harrison, R., (2016) *"A lightweight approach for human factor assessment in virtual assembly designs: an evaluation model for postural risk and metabolic workload",* Proceedings of the 6th CIRP Conference on Assembly Technologies and Systems (CATS), 16-18th May 2016, Gothenburg, Sweden

Conference Presentations (slideshows)

- *"Connecting Assembly Process Planning with Machine Control Software",* WMG Doctoral Research and Innovation Conference, University of Warwick, 28th June 2017
- *"Automating Automation – Machine Sequence Changes Using Ontologies",* Research Student Skills Programme (RSSP) Poster competition, University of Warwick, 2016
- *"A PEM fuel cell ontology to facilitate assembly line generation using a semantic approach: A proof of concept",* Fuel Cell and Hydrogen Technical Conference 2015 & WMG Doctoral Research and Innovation Conference, 30th June 2015. **BEST PAPER AWARD**
- *"Fuel Cell Pick and Place Assembly System Complexity",* Midlands Energy Consortium (MEC) Student Conference, 17th Dec 2015
- *"Finding an optimal fuel cell assembly sequence",* Hydrogen and Fuel Cell Research Conference, Bath, 2015
- *"Developing a PEM fuel cell master assembly sequence",* Future Powertrains Conference (FPC), 2015 - Poster
- *"Benchmarking the fuel cell compression process for the Horizon Closed Cathode Fuel Cell Stack using Fuji Prescale pressure sensitive films",* Hydrogen and Fuel Cell Research Conference, 15-17th Dec, 2014.

Acknowledgements

There are a number of individuals I would like to thank for their support and contributions without whom this work would not have come to fruition.

Firstly, I would like to thank my main supervisor, Robert Harrison, who is a true role model for integrity and humility. He believed in me and my work when I did not and his door was always open. I am both honoured and grateful to have had the opportunity to have worked with him and look forward to continuing to do so. I would also like to thank Bilal Ahmad for his supervision and support, as well as my external supervisors James Meredith and Axel Bindel. I must also extend my sincerest thanks to the High Speed Sustainable Institute (HSSMI) and the Engineering and Physical Sciences Research Council (EPSRC) for funding my research work through the Knowledge Driven Configurable Manufacturing (KDCM) project as well as the associated iCASE studentship.

All of my colleagues at the Automation Systems Group at the University of Warwick have been a source of inspiration, in particular Daniel Vera, Karthik Ananthanaryan, and Bugra Alkan, all who have provided invaluable friendship and advice. In addition, I must thank my friend Manal Assaad has been instrumental in providing support and motivation as I have worked through the thesis correction process.

I must also extend my thanks to my good friend and colleague at the FAST lab at the Technical University of Tampere, Borja Ramis Ferrer. Without his support and contribution, I would not have been able to do this work. In addition, I am grateful for the technical support provided by my colleague Jiayi Zhang when my programming skills failed me.

A big thanks also goes to colleagues at the various companies that I have interacted with over the course of my research who have made their facilities and knowledge open and available to me, in particular Ben Todd and Jermu Pulli at Arcola Energy, Daniel Bolton at Horizon Instruments, and Tim Nixon at Contron.

I wish to express my sincerest gratitude to my parents who have been incredibly patient, supportive, and accommodating through this journey. The English vocabulary is insufficiently expressive to describe how grateful I am to you both for everything.

Finally and most importantly, I thank God. He has given me opportunities I could never have dreamed of and bequeathed me with the capability to experience the world.

However, fleeting it may be, it is certainly exciting to exist.

Contents

Table of Figures

Table of Tables

Abbreviations and Acronyms

ACO	Ant Colony Optimisation
ADACOR	ADAptive holonic Control aRchitecture for distributed manufacturing control
AGV	Automated Guided Vehicles
ALB	Assembly Line Balancing
API	Application Programming Interface
APP	Assembly Process Planning
ASP	Assembly Sequence Planning
BFO	Basic Formal Ontology
BOM	Bill of Materials
BOP	Bill of Process
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAPP	Computer Aided Process Planning
CCS	Capability Class Structure
CPPS	Cyber Physical Production System
CPS	Cyber Physical System
DFA	Design For Assembly
DFMEA	Design Failure Modes Effects and Analysis
DRM	Design Research Methodology
EAS	Evolvable Assembly Systems
EC	Engineering Change
ECM	Engineering Change management
FBS	Function Behaviour Structure
FMS	Flexible Manufacturing System
GA	Genetic Algorithm
GATE	General Architecture for Text Engineering
GD&T	Geometric Dimensioning and Tolerancing
GDL	Gas Diffusion Layer
GUI	Graphical User Interface
HMS	Holonic Manufacturing System
ID	Identification Number
IDEAS	Instantly Deployable Evolvable Assembly Systems
ISO	International Organization for Standardization

JAPE	Java Annotation Pattern Engine
KB	Knowledge Base
KDCM	Knowledge Driven Configurable Manufacturing
KIF	Knowledge Integration Framework
KR	Knowledge Representation
MBE	Model Based Engineering
MBSE	Model Based Systems Engineering
MCCO	Manufacturing Core Concepts Ontology
MCM	Manufacturing Change Management
MCS	Machine Control Software
MDD	Manufacturing Domain Data Model
MDM	Manufacturing Data Model
MODAPTS	MODular Arrangement of Predetermined Time Standards
MSE	Manufacturing Systems Engineering
MSU	Manufacturing Software Unit
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology
OCL	Object Constraint Language
ONTOMAS	Ontology for the design of Modular Assembly Systems
OPC-UA	Open Platform Communications Unified Architecture
OWL	Web Ontology Language
PDM	Product Data Management
PEM	Proton Exchange Membrane
PERFoRM	Production harmonized Reconfiguration of Flexible Robots and Machinery
PFMEA	Process Failure Modes Effects and Analysis
PLC	Programmable Logic Controller
PLM	Product Lifecycle Management
PMI	Product Manufacturing Information
PMTS	Predetermined Motion Time System
PPR	Product, Process, Resource
PSL	Process Specification Language
PSO	Particle Swarm Optimisation
RDF	Resource Description Framework
RMS	Reconfigurable Manufacturing System

ROSETTA	Robot control for Skilled ExecuTion of Tasks in natural interaction with humans; based on Autonomy, cumulative knowledge and learning
SA	Simulated Annealing
SFC	Sequential Function Chart
SIARAS	Skill-based Inspection and Assembly for Reconfigurable Automation Systems
SIC	Sequence Interlock Chart
SkillPro	Skill-based Propagation of "Plug&Produce"-Devices in Reconfigurable Production Systems by AML
SOP	Sequence of Operations
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SWRL	Semantic Web Rule language
SWT	Semantic Web Technologies
SysML	System Modelling Language
TC	Timing Chart
UML	Unified Modelling Language
URI	Unique Resource Identifier
VE	Virtual Engineering
VFF	Virtual Factory Framework
V-Man	Virtual Manikin
V-Rob	Virtual Robot
W3C	World Wide Web Consortium
XML	Extensible Markup Language

1 Introduction

1.1 *Problem Background*

1.1.1 Industrial Challenges

Manufacturing organisations face a number of challenges in the modern age. The emergence of smaller electronics, more powerful hardware, and better access to software has resulted in the evolution of complex, high value products across a range of industries. Many products now consist of their mechanical arrangement, electrical system, and software implementation. Furthermore, a more demanding customer base has resulted in a paradigm shift away from mass production and through to mass customisation. This is prevalent in the automotive sector where the number of product variants continue to increase as manufacturers rapidly approach a “batch size of one” (ElMaraghy, 2012). It has been reported that the number of product variants has increased between 500% and 700% in the German industry while production volumes have dropped to only as little as 85% of their original volume (ElMaraghy, 2012). This turbulent environment calls organisations to increase their responsiveness to maintain productivity and prevent costs from escalating to a point where they are no longer competitive.

In order to meet safety, quality, and volume requirements, industrial automation systems are often employed within manufacturing organisations. The workflow to realise an industrial automation system is complex and is presented in Figure 1-1. The complexity associated with realising industrial automation systems is attributed to in part the number of stakeholders involved and also the number of times information is exchanged as a consequence. Managing and controlling the exchange of information and ensuring that requirements and challenges are being correctly communicated is an on-going battle in a manufacturing organisation that employs industrial automation systems. Furthermore, the engineering and development processes for such organisations remains ad-hoc and unconnected across phases and domains. Therefore, there is an opportunity to learn from the shortcomings of existing product development processes and manufacturing system data integration methods.

The workflow presented in Figure 1-1 is summarised as follows, assuming the position that an ***assembly*** automation system is the objective:

- A new product or product variant is introduced that is designed by the product designer. There is typically a process of prototyping the product within both simulation environments through finite element modelling by product engineers. Furthermore, there may also be some work done within design teams to consider design for assembly or design for manufacture with some proposals for the sequence and nature of processes
- The Product Domain then exports information concerning a product which is continuously changing as the product enters different stages of maturity from conceptual through to final design
- The Process Domain imports some of the product information and combined with knowledge about manufacturing processes, including those capabilities that exist within the organisation and also what other processes exist beyond this, the process planner transforms product design information into an assembly process plan.
- The Process Domain also exchanges information with the Product Domain about what constraints surround the product in terms of design for assembly and in some cases some product design changes are executed as a direct consequence of feedback
- The information exported from the Product Domain and the Process Domain is then absorbed by the Resource Domain which is initiated through interaction with a machine builder, who is typically trained as a mechanical engineer. The mechanical engineer's role in this context is to transform the combination of product and process requirements into the instantiation of a physical machine
- The mechanical engineer begins a conceptualisation process considering the mechanical arrangement of components and the types of automation components that will be used based on the requirements
- This conceptualisation is fed to both the controls engineering and the process planner in the form a timing diagram which illustrates the machines behaviour
- The process planner compared the timing diagram with the assembly process plan to ensure requirements are met and criticise design proposals accordingly
- In the same way, the controls engineer creates a sequence interlock chart which describes how the proposed behavioural requirements are to be transformed into machine control code with the appropriate sequence checks and interlocks. This chart is compared with the timing diagram to ensure requirements are being met

- The machine design generated by the mechanical engineer is fed to the electrical engineer who then begins the process of design the electrical system and eventually constructing the electrical cabinet
- The mechanical engineer instantiates the building the machine, the controls engineer programs it
- Ultimately a physical machine is built by the Resource Domain that is able to automate the assembly process envisioned by the Process Domain, which realises a product created by the Product Domain.

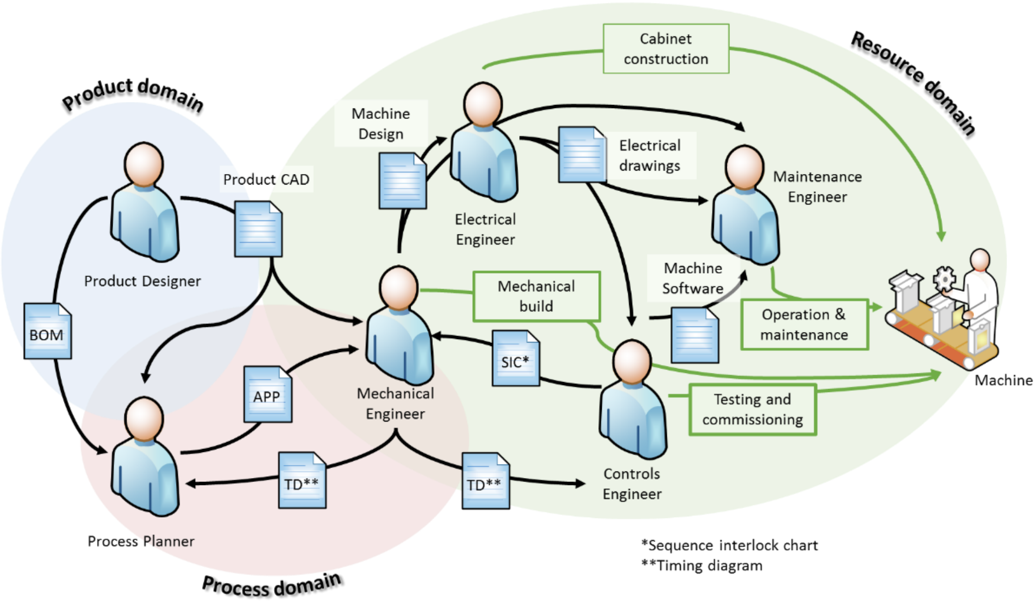


Figure 1-1 Workflow to realising industrial automation systems

An additional challenge that manufacturers must contend with is that of sustainability. Ever stringent government legislation impose significant pressure on companies to work towards holistic strategies that develop the pillars of sustainability: society, environment, and economy (Bi, 2011). All three pillars have much to gain from the transition, or at least the development of, technologies that consume renewable energy, use energy more efficiently, and with less harmful waste products. Within the context of power generation technologies such as combustion engines, a clear motivation exists to move towards cleaner and “greener” offerings. However, there is a significant lack of knowledge associated with the manufacturing and assembly of these products that include: electric machines for propulsion, battery packs for energy storage, and hydrogen fuel cells as a potential direct replacement of the internal combustion engine (Wang *et al.*, 2013, Mehta and Cooper, 2003, Çağatay Bayindir *et al.*, 2011, Sharaf and Orhan, 2014). It is important to note that these technologies are not limited to automotive or transport. Distributed, off-grid, backup and portable power all offer opportunities for these new technologies (Sharaf and Orhan, 2014). However, the in-house knowledge that exists within the organisation of those manufacturers that are experts in producing conventional technologies, does not extend to producing the aforementioned set of sustainable technologies. With engineering workflows that rely on the knowledge of experts, introducing and developing entirely new products presents a significant challenge. To address the concerns and risks related to climate change, conventional workflows cannot be relied upon to realise the aforementioned new technologies in a timely manner.

In order to develop new technologies in a rapidly changing environment, effective knowledge and information management is essential. Understanding the interaction between product requirements and manufacturing system capability is a fundamental part of this. The ability to know where and how a manufacturing system must adapt as a consequence of new product design requirements requires the connection of multiple engineering and design disciplines. Current industrial data management tools are not sufficiently well integrated or provide sufficient detail for users to fully appreciate the impact of change. Furthermore, design and engineering tools across the domains of product realisation (product, process, and resource) suffer from poor interoperability i.e. models are created in proprietary standards, utilise inconsistent semantic descriptions across domains, and there is a lack of model maintenance preventing digitisation efforts to be fully exploited (Harrison;, 2017, Harrison *et al.*, 2016). Finally, due to the dynamic nature of production environments, tools and methods originally developed for mass production scenarios are unable to react with the necessary agility (Järvenpää, 2012).

1.1.2 Scientific Challenges

The academic community are well aware of the challenges facing industry solutions have been presented to address them. A broad variety of manufacturing paradigms have been introduced, developed and tested with varying degrees of success. Flexible manufacturing system (FMS) are able to meet a broad range of requirements without modifications to the system structure (Hu *et al.*, 2011, ElMaraghy *et al.*, 2013, ElMaraghy and Wiendahl, 2014). On the other hand reconfigurable manufacturing system (RMS) adapt to change through modification of the manufacturing system’s mechanical or software components (Järvenpää *et al.*, 2016, Koren and Shpitalni, 2010). This vision requires the manufacturing system to have certain characteristics such as modularity and convertibility (Koren and Shpitalni, 2010). As the computing power of control and controlled devices has increased, the paradigm of agent-based and holonic manufacturing systems (HMS) has emerged (Leitão, 2009). This paradigm focuses largely on the software aspect of change, enabling dynamic shop floor behaviour adaption through intelligent, autonomous entities that can interact with their environment and other agents to achieve a goal (Leitão, 2009).

The vision presented by the Industrie 4.0 framework is one of self-organization, self-optimisation, and self-diagnosis, with a view to achieving a business goal (Westkämper and Jendoubi, 2003, Adolphs *et al.*, 2015, Hankel and Rexroth, 2015).This vision is enabled through some mix of the aforementioned paradigms. This is supplemented by more effective communication and integration of domains of engineering and data/information models across the lifecycle and through the business.

However, regardless of what combination of technologies and paradigms for the required degree of dynamism are implemented on the shop floor (FMS, RMS, HMS), it is necessary for such changes to be managed and executed through domains of design and engineering that sit outside of the factory. In other words it is not enough to develop hardware, software, and engineering tools and methods that enable the factory to change. Currently, there is limited research on considering how product and process information evolves and finds itself in the factory's domain (Järvenpää *et al.*, 2016, Järvenpää, 2012). The understanding of this would allow the development of more formal links within and across the product realisation domains. This in turn facilitates the modelling and thus predictions of the impact and/or nature of change on the factory. Appropriate preparations for change can be made and the relevant engineering teams have more time to find and develop optimal solutions than what is often a last minute, ad-hoc and often expensive approach. There is a lack of research that formalises knowledge associated with what the factory is able to do and linking this with what it is required to do. Formalisation of this nature would enable changes to be executed more successfully across the domains and maintain a structured engineering workflow through the lifecycle of both the product and the manufacturing system.

The challenge presented to the scientific community is either to: i) develop and test new engineering tools and methods that are at their onset open and use non-proprietary data in the hope that industry engages with such solutions to encourage further funding or ii) develop tools and methods that store knowledge for integrating pre-existing tools and methods through neutral exchange formats hoping that the concepts are sufficiently comprehensive and generic to accommodate the breadth of complexity required.

1.1.3 Summary

The industry needs to work towards evolving engineering workflows such that it is more readily able to adapt to change as a consequence of new product variants, reduced product lifecycle, mass customisation, and governmental and consumer pressures to transition towards more sustainable products and manufacturing systems. The scientific community needs to develop methodologies that enable this requirement while complementing existing approaches, tools, and methods. It is clear that the digitisation of products and factories is an enabler of more effective data integration, but to truly exploit this paradigm, transforming data into knowledge that can be reused to make informed decisions is key. Figure 1-2 illustrates a high level view of the problem and shows that while there is some integration or communication of information at the tool layer across domains i.e. some parameters or pieces of information can be communicated from one piece of software to another, the respective data or information models are not effectively integrated.

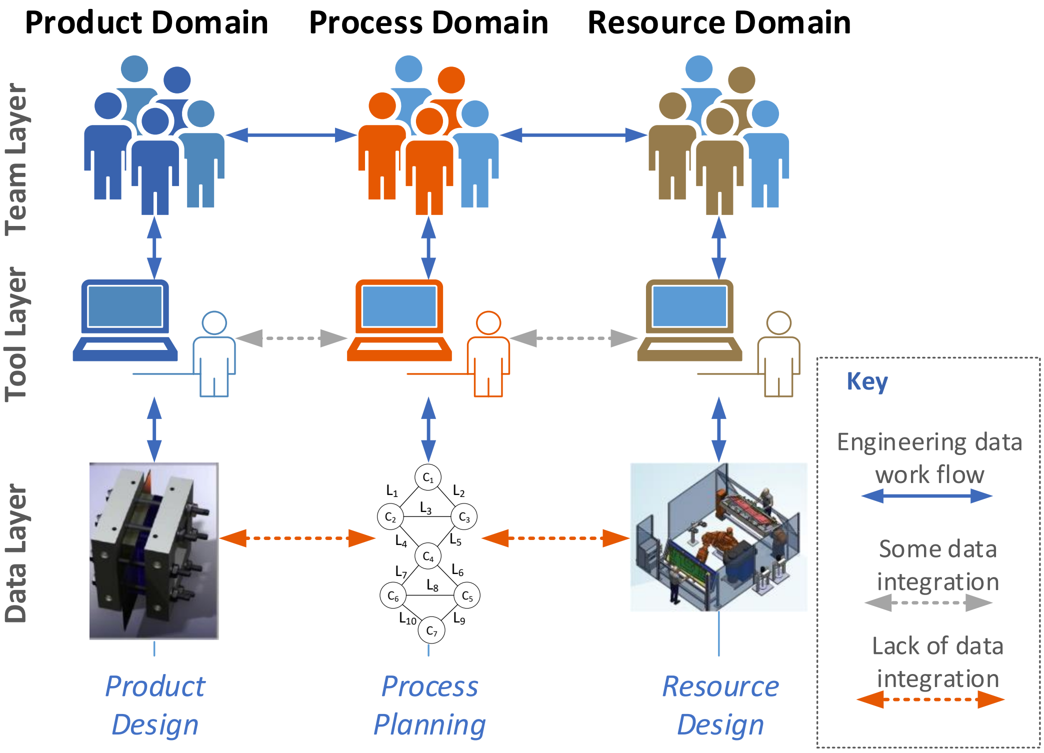


Figure 1-2 Problem Background

1.2 *Formulation of Research Problem*

1.2.1 Vision

The wider context of this work is a vision whereby the engineering workflow through the product realisation domains is streamlined, efficient, and error free. The transition from data and information towards knowledge is to be enabled by better integrated engineering tools allowing better informed decisions to be made more quickly.

It is envisioned that designers and engineers will be focused on innovation and adding value to a business rather than engaging with inconsistencies, model discrepancies, and miscommunication due to a lack of domain integration. When changes are made in one domain the relevant stakeholders are advised and appropriate solutions can be selected from a knowledge base.

Solutions within engineering and manufacturing organisations often need to be bespoke to account for nuances and subtleties associated with small details that are overlooked during the modelling process. It is the resolution of these nuances in an innovative way that should be the focus of those employed by such organisations. Effective information model integration coupled with a knowledge-base is a key enabler of this.

Moving into the operational phase of a manufacturing system, it is envisioned that changes associated with the logical aspects of process e.g. sequence, and in the future the mechanical arrangement of equipment could be reconfigured through standard-driven auto-code generation and technologies such as AGVs (Automated Guided Vehicles). This would be achieved through knowledge-models that infer the new requirements based on knowledge about exist capabilities and limitations. Automatic notification of where shortcomings may exist within shop floor component libraries would be highlighted to the relevant stakeholders rather than such people actively having to sift through complex models and large amounts of paperwork to determine where limitations exist.

Ultimately, the increased responsiveness and ease of implementing change will free up resources such that new products are more innovative increasing the value of the engineering process. Less time is spent on administration and non-value adding work. In addition, the quality of products is expected to increase as potential shortcomings associated with inadequate processes can be more readily identified.

1.2.2 Problem Synthesis

As discussed in Section 1.1.1 the current method for communicating the requirements of the product and process through to machine design and control remains either document based or model-based with poor model coupling, despite the respective activities utilising their own engineering tools. This means that documents or models need to be examined and interrogated to extract the key information proceed to the next stage of the product realisation process. This also means that when changes are made, the manual interrogation process needs to be carried out to identify where the changes have been made and what impact they will have.

In a model-based approach it is often necessary to develop software plugins through APIs (application program interface) through languages such as Visual Basic (Balena and Foreword By-Fawcette, 1999) to connect models in an ad hoc way and, due to the lack of semantic formalisation, model modifications render plugins unusable. The efforts of interrogation, correction, modification and the like results in high costs and prolonged lead times which negatively impact on the organisation's ability to meet the needs of the customer i.e. customisation.

The use of knowledge representation (KR) through methods like ontologies has seen limited to no use in the manufacturing industry. However, the awareness of such technologies is growing which is a direct consequence of the increasing number of large EU projects that bring together academic expertise and industrial state of the art. Despite these efforts, the problem remains that ontologies are powerful tools within their own right, but it is not clear how they can be used to support the engineering activities associated with product realisation with a view to supporting product design, process planning, and machine reconfiguration.

Due to the breadth of changes that a manufacturer can face as an outcome of introducing a new product, the author chooses to focus on the specific problem associated with control logic. Changes within the manufacturing system can be classified into physical and software based. Physical changes encompass those of a mechanical nature which form the majority, but can also include electrical.

On the other hand, software changes are those associated with parametric changes and those that are logic based. When introducing a new product or product variant, there is a risk that the assembly sequence and in turn the control logic of the automated machine will change also. Although the work is proposes a method to identify the consistency between the mechanical nature of a given machine and the mechanical requirements of a product (as per the Skill model in the vision section), the primary focus remains on the software side.

1.2.3 Research aim and hypothesis

The core aim of this research is to understand why ontologies (or other forms of knowledge representation) are not being more extensively used beyond academic settings, particularly in manufacturing environments. Based on this understanding, the author aims to develop a toolset or workflow that supports an engineering process that is beyond the capability of tools and methods that are classically deployed in industrial settings e.g. Product Lifecycle Management through relational databases.

The hypothesis of this research is therefore that:

"Ontologies can be integrated with engineering tools to complement existing engineering workflows through the identification and resolution of inconsistencies between typically un-integrated and disparate engineering models, complementing and enhancing the capability of databases."

1.2.4 Research questions and objectives

Based on the problems identified, the research aim and the research hypothesis, the following questions are raised:

1. How is change management executed within industrial settings and what methodologies have evolved to support this process?
2. What are the shortcomings of existing ontological models that prevent their use within industrial settings to support change management?
3. How can ontologies be used in conjunction with engineering tools and methods to complement existing engineering workflows to support the introduction of new products?
4. How can assembly process plans be connected to machine control software through ontological models to facilitate in the identification of inconsistencies with a view to resolving them?

The following objectives are derived from the questions above:

1. Identify change management methods within the context of manufacturing and engineering changes and the challenges that are faced

2. Identify the ontological models that have been developed in the literature and how they have been applied as well as their shortcomings
3. Develop a set of PPR ontologies that can be used to support assembly automation systems engineering through its lifecycle
4. Develop a framework that integrates engineering tools, methods, and workflows with an ontological model
5. Demonstrate how ontologies can be used in a practical way to identify and resolve inconsistencies

1.2.5 Scope – Limitations and Assumptions

The domain of manufacturing includes activities and processes that range from the creation of components from raw materials, the transformation of material properties through a range of tightly controlled processes, as well as the assembly of components to produce sub-assemblies or products. In order to keep the scope of this research within that of a PhD thesis, this work focuses only on assembly of products, assembly processes, and assembly systems. However, the modelling approach used in this thesis could be extended to include manufacturing processes beyond assembly. Furthermore, the modelling of semi-automated systems i.e. those where human-machine interaction exists, is not within scope of this research. Again, the modelling approach could potentially be extended to support this as the virtual engineering tools used have functionality to model human-machine interaction.

The work in this thesis has complemented an Innovate UK project titled Fuel Cell Manufacturing and the Supply Chain (project ref: 101980) and more recently DIGIMAN (DIGItal MAterials CharacterisatioN proof-of-process auto assembly) which is a Horizon 2020 project funded by the European Commission (project ref: 736290). Both projects investigate the challenges associated with fuel cell manufacturing and assembly with the former focusing on low volume production (up to 1000 stacks per year) and the latter on mid-high volume production (up to 100,000 stacks per year). This thesis captures some of the challenges that have been identified with fuel cell assembly systems. The problems have been abstracted into more general ones that could also be linked to other similar products such as battery packs. In addition, the Knowledge Driven Configurable Manufacturing (KDCM) (EP/K018191/1) project has been vital in funding the development of virtual engineering tools that build upon the component paradigm. These tools have been used extensively in the case study chapter (Chapter 4).

1.2.6 Contributions

There are three contributions of this thesis. The justification that these gaps exist within the body of knowledge is made in the literature review (Chapter 2) and the verification that they have been addressed is made through case studies and their evaluation. The contributions are summarised as follows, with more detail presented in the concluding chapter:

1. **Development of PPR Ontologies and a Skill Model** to address the lack of explicitly defined ontological models that describe the concepts and relations across the two
2. **The development of a broader framework for integrating ontologies with virtual engineering tools** to demonstrate how ontological models can be used in conjunction with engineering tools and thus complement existing engineering workflows
3. **The use of ontological models to support inconsistency management** to build a stronger case for their use in engineering settings and complement existing data storage methods e.g. relational databases within Product Lifecycle Management systems.

In summary, the thesis identifies the shortcomings of existing ontological models within the context of manufacturing, develops new models to address those shortcoming, and develops new, useful ways for ontological models to be used to address industrial problems by integrating them with virtual engineering tools.

1.3 *Research Methodology*

The research methodology adopted for this research originates from design science. In this research, the Design Research Methodology (DRM) is used (Blessing and Chakrabarti, 2009). This is the same methodology used by Järvenpää (Järvenpää, 2012) due to the similarity of the field and approach used. The methodology is split into four stages which are described as follows:

- **Criteria Formulation:** identification of the aim of the thesis which is to identify a means to integrate engineering data through the lifecycle and across domains. This is to support industry which is engaged with tackling the problem of product variety, mass customisation, as well as the more broad environmental challenges generating a need for alternative energy technologies. Conventionally, criteria formulation would present criteria that are measurable e.g. increased profitability of an organisation to illustrate the efficacy or benefits of a method. However, due to practical constraints associated with the time-scale of the research project and the availability of data, it is not possible to assess whether the methodology presented in this research is measurably better than industrial state of the art. As a result, the work is assessed based on the approaches identified in scientific literature and how the methodology in this research extends them and fills current gaps.

- **Descriptive Study I:** the role of this stage is to increase the understanding of the existing methods and tools in industry and academia to identify current practices and thus a basis on which improvements can be made. As described by Blessing and Chakrabarti (Blessing and Chakrabarti, 2009) this generates a reference model. Within the context of this thesis, the outcome of this stage is to shed light on the existing workflow and associated tools. This is achieved through a literature review (see Chapter 2) as well as the experience of the author's involvement with industrial research projects and the insights acquired due to this exposure.
- **Prescriptive Study:** the outcome of the Descriptive Study is used to identify a reference model or theory that the Prescriptive study extends or develops. In the case of this research thesis, the Descriptive Study (literature review) is used to build a PPR ontology in conjunction with a Skill model with concepts and relations derived and inspired by what already exists, but then extended to address what doesn't. In addition, gaps in the knowledge concerning broader frameworks that include ontologies within engineering workflows are created. The methodology is then tested and validate through case studies in Chapter 4.
- **Descriptive Study II:** a second descriptive study is undertaken to evaluate the application of the methodology developed in the Prescriptive Study. The evaluation considers both the *application* to identify whether the method has the expected effect and the *success* to determine whether the method is beneficial. The *application evaluation* is carried out within the Case Study chapter while the *success evaluation* is carried out in the Evaluation chapter.

1.4 *Thesis Outline*

The thesis is organised as follows:

- **Chapter 2 Literature Survey** – A review of the literature in the areas of: ontological models in manufacturing, skill and capability modelling and their uses, inconsistency and change management in manufacturing, state of the art in the management of product data in industry such as PLM. The chapter concludes with a summary clearly identifying the gaps in the literature.

- **Chapter 3 A knowledge-based approach for integrating engineering workflows** – Based on the gaps identified in the literature review, a PPR ontology with a Skill model is created and it is shown how the authors envision such a model would integrate with the engineering workflow to realise and support manufacturing. The focus of the model is on assembly, however due to its modular nature, it could be extended to encompass other domains of process activities and the associated manufacturing resources.
- **Chapter 4 Case Study –** To test the models developed in the methodology chapter three case studies are presented. The first demonstrates how the Skill model would enable the verification that the skills or capabilities required of the manufacturing system exist within it. The second demonstrates how the identification of an inconsistency from a logical perspective, that is to say that the process plan is inconsistent with a piece of automation equipment's control logic, is identified. The third case study validates the resolution of inconsistencies. All case studies are demonstrated through virtual modelling as it is important to show how engineering tools integrate with the proposed knowledge-based approach.
- **Chapter 5 Discussion and Evaluation** – Based on the results of the case studies, the models generated are evaluated as well as the approach more broadly. This is carried out in a qualitative way by revisiting the literature review and the gaps identified and checking how the research addresses them. This analysis justifies what the author argues to be the contribution of the work. In addition, the models and method is critiqued to extract the shortcomings as a basis for future work that should be done.
- **Chapter 6 Conclusion and Further Work** – The work as a whole is summarised and remarks are made with respect to the problems identified in Chapter 1 discussing the impact of the work both within an industrial and scientific context. Finally, the contributions of the work are also summarised with future research questions and directions generated as a consequence of this research project.

The DRM has been used as a supporting tool to structure this research work and ensure a certain level of academic rigour. Figure 1-3 illustrates how the thesis structure presented in this section aligns with the DRM discussed in Section 3.

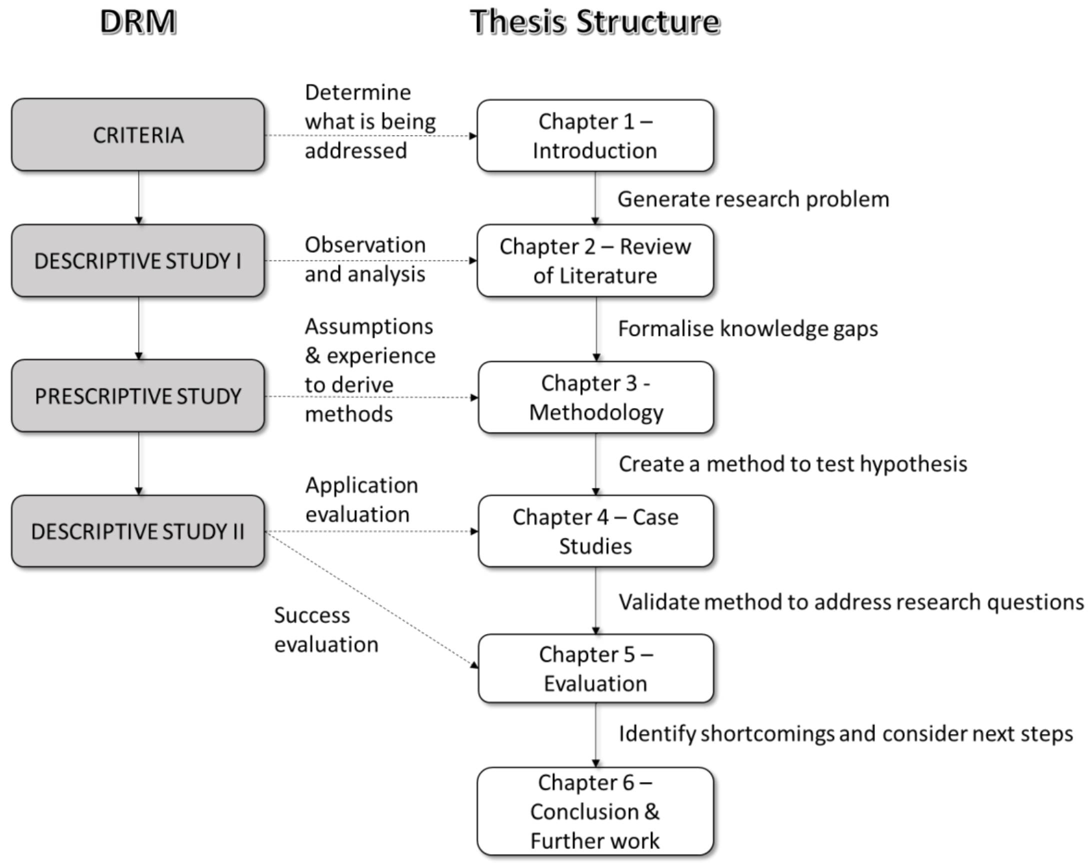

DRM
Thesis Structure
CRITERIA
Determine what is being addressed
Chapter 1 – Introduction
Generate research problem
DESCRIPTIVE STUDY I
Observation and analysis
Chapter 2 – Review of Literature
Formalise knowledge gaps
PRESCRIPTIVE STUDY
Assumptions & experience to derive methods
Chapter 3 - Methodology
Create a method to test hypothesis
DESCRIPTIVE STUDY II
Application evaluation
Chapter 4 – Case Studies
Success evaluation
Validate method to address research questions
Chapter 5 – Evaluation
Identify shortcomings and consider next steps
Chapter 6 – Conclusion & Further work

Figure 1-3 PhD thesis structure aligned with DRM framework

2 Literature Review

2.1 *Introduction*

This chapter identifies and reviews literature within the context of methods and tools for integrating the lifecycle data associated with manufacturing systems. The manufacturing system in this review is defined by the aggregation of the engineering and design activities in the Product, Process, and Resource (PPR) domains. More specifically, the review focuses on assembly, but relevant tools and methods beyond this domain are referenced also. Each of the PPR domains are highly interconnected and the nature of their characteristics and interaction is fundamental in understanding how best to integrate them.

This chapter opens with describing change propagation and the engineering change management process presenting the challenges highlighted in the literature. This is reviewed as the paradigm shift towards mass customisation, personalised products, and reduced product lifecycles necessitates more agility on the part of the organisation (Mourtzis and Doukas, 2014). It is important to understand how this is currently managed in industry and academia to identify the shortcomings and address them accordingly.

One of the major enablers of managing change is the transition from document-based engineering to model-based engineering (MBE) which moves the record of authority from documents to digital models. This allows engineering teams to more readily understand design change impacts, communicate design intent, and analyse a system's design prior to build (Hart, 2015). Model based systems engineering (MBSE) is a focused version of MBE specifically associated with systems engineering. The literature review focuses on the use of MBSE within the context of the manufacturing workflow i.e. PLM, and what methodologies exist to link the aforementioned PPR domains, highlighting the limited use of knowledge management within this context.

Next, a review of tools and methods associated with assembly process planning (APP) is presented. This section of the literature is presented as assembly process planning is an activity which connects the Product and Resource domains. It is therefore considered that the Process domain is defined at least by the definition of APP if PPR exists within the domain of assembly. However, the author reiterates that within the broader scope of manufacturing there is the activity of "process planning" which would encompass processes beyond the scope of assembly e.g. casting, milling, extruding etc.

Finally, as the methodology utilised in this thesis to demonstrate how the PPR domains can be connected in an intelligent way uses ontological models, the review identifies the existing use of ontologies in the literature within the context of manufacturing. This section of the review also examines the use of skill or capability models and how they integrate with or are used in a complementary way with PPR models.

2.2 *Engineering Change Management: the process and the challenges*

The term engineering change (EC) has been defined in multiple ways and has continued to evolve as the complexity associated with the process and its multidisciplinary nature has increased. In a review of engineering change management published in 2012, the following definition was proposed: "ECs are changes and/or modifications to released structure, behaviour, function, or the relations between functions and behaviours of a technical artefact (Hamraz *et al.*, 2013)." The key message to take away from this definition is that the engineering change is to be effected upon an artefact that exists or has been "released." This does not mean that engineering changes do not occur during an initial design process i.e. original designs during the product development phase (Otto and Wood, 1998, Pahl and Beitz, 2013). However, it is agreed that the majority of engineering changes occur as a consequence of evolutionary design (Bentley and Corne, 2002, Kicinger *et al.*, 2005).

The management of ECs requires the investment of significant resources in manufacturing organisation (Huang and Mak, 1999). In the automotive industry, Ford, GM and DaimlerChrysler claimed that they handled approximately 350,000 ECs in a single year combined. The organisations suggested that the costs per change were more than $50,000, including both the capital investment involved as well as lost man hours and delays (Wasmer *et al.*, 2011). It is argued that design changes that occur later in the development lifecycle can be up to 10 times more costly to implement than those identified at earlier stages (E. Carter and S. Baker, 1992). Whether the cost is as great as suggested by Carter and Baker (E. Carter and S. Baker, 1992) is debatable, however other literature does suggest figures that do show significant cost differences at different lifecycle stages e.g. prototype phase: <$20,000 and after production: >$100,000 (Stamatis, 2002). However, it is necessary to be able to make changes even when a manufacturing system is mature as it is at this stage of its lifecycle that it exists for the longest amount of time. In fact, due to this idea, the paradigms of flexible manufacturing systems (FMS) and reconfigurable manufacturing systems (RMS) emerged to be able to accommodate change during the operational phase of a manufacturing system (ElMaraghy, 2006, Koren *et al.*, 1999, Hu *et al.*, 2011, Tolio *et al.*, 2010).

2.2.1 Steps and Methods for Executing Engineering Change Management

Engineering change management (ECM) can be regarded as the core function of configuration management, which in turn is a discipline of systems engineering (Jarratt *et al.*, 2011). In a review on the topic of engineering change management (ECM) in 1996, Huang and Mak (Huang and Mak, 1999) found that two main approaches existed: formal and ad hoc. The former approach was used by approximately 95 per cent of those that responded to the questionnaire that was sent out as part of the review. The authors had usable data from approximately 100 companies, from an initial sample size of 2,000. Of those that did not respond the reasons included a lack of relevance of ECM to their business, or a lack of necessary data to complete the questionnaire. As the questionnaire was sent only to companies that were involved in product design and manufacture, the author's conclusion that a high percentage of companies utilise formal ECM processes is not fully justified. It seems that only those that did have a formal ECM process responded, from this it could be inferred that the majority of companies in fact use ad hoc methods. This view is supported by similar studies at the time Huang and Mak (Huang and Mak, 1999) published their work e.g. Maull *et al.* (Maull *et al.*, 1992).

Within the context of formal methods, the set of steps identified for executing ECs in Huang and Mak (Huang and Mak, 1999) align with more recent studies (Jarratt *et al.*, 2011, Quintana *et al.*, 2012, Hayes, 2014). These can be summarised as: 1) raise an engineering change request which contains reasoning, instructions, and risk assessment, 2) a decision is made as to whether the request should be actioned, 3) the change is executed or implemented, and finally 4) documentation pertaining to the change is updated and reviewed. The level of detail varies depending on the study, but the sequence of steps remains consistent across the literature.

The consensus regarding formal methods for ECM is that traditional paper-based approaches are not suitable and thus digital methods have been developed (Demoly *et al.*, 2013, Do, 2015, Wasmer *et al.*, 2011). These largely integrate with or are synonymous to product lifecycle management (PLM) or product data management (PDM) i.e. Wasmer *et al.* (Wasmer *et al.*, 2011, Do, 2015). Huang *et al.* (Huang *et al.*, 2003) describe a set of standardised frameworks for change management: MIL-STD-973, ISO 9000, ISO 10007, and BS 6488:1984. However, they found limited evidence for their regular and formal use in industrial applications. The Institute of Configuration Management (2004) developed the CMII model as framework to support products, processes, and facilities by managing their associated information. According to (Wu *et al.*, 2012) however there has been limited implementation of the model within an industrial context and thus they attempt to apply it to the manufacture of motorcycles by creating Full-track and Fast-track EC workflows. However, the integration of the proposed framework within existing engineering methods and tools is not described

In addition to software tools and standards, implementation of engineering changes are supported by specially appointed personnel within an organisation referred to an EC co-ordinator or an EC board (Huang and Mak, 1999). These people operate between respective disciplines or departments to communicate changes, align stakeholders, and also make decisions between the key EC steps. However, these systems and methods remain largely focused on product data and thus *product* focused engineering changes (Shankar *et al.*, 2012). In fact, much of the literature on ECM typically focuses on the Product domain, considering process changes and manufacturing resource changes simply a consequence (Hamraz *et al.*, 2013, Koch *et al.*, 2016) even though these activities could still be considered within the definition of the “manufacturing system”. This view is a consensus, evolving from the work of McMahon in 1994 (Hamraz *et al.*, 2013, McMahon, 1994).

2.2.2 Manufacturing Change Management

A recent study by (Koch *et al.*, 2016) highlighted the issue of the lack of literature concerning the management of change within the area of manufacturing. To that effect, they created and validated a manufacturing change management (MCM) process that they derived from the literature within the areas of: MCM, factory planning, continuous manufacturing planning, and ECM. They concluded that the following processes defined the MCM: need for change, change identification, solution finding, evaluation and decision, change planning, implementation planning, implementation, knowledge management and control. The author notes that this differs very little from the literature derived stages associated with ECM (Figure 2-1). This highlights that regardless of the domain, the process steps associated with change are similar if not the same. Regardless, the management of engineering changes remains disjointed and document-based or through the use of software tools that digitise the process.

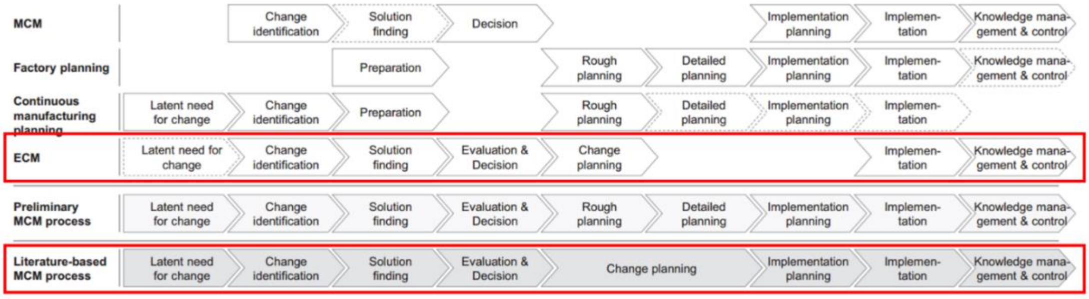


Figure 2-1 Commonality between ECM and MCM modified from Koch et al. (Koch et al., 2016) indicating that ECM, when considered from a process perspective, is very similar to MCM.

2.2.3 Research Opportunities in Engineering Change Management

Section 2 has touched upon the steps required to execute an engineering change and highlighted some of the methods to do this e.g. software tools, standards, and elected personnel. Furthermore, the limited consideration for engineering changes outside of the product domain highlights a gap in ECM literature. The caveat here is that in some sense, resource components i.e. elements of a manufacturing system that perform process tasks to realise a product (Labrousse and Bernard, 2008), could be considered a Product i.e. an element which is at the first instance designed for a given requirement and produced accordingly, at some point in their lifecycle. However, even given this, the interaction and evolution of such entities is not the focus of ECM. Some simulations have been developed which investigate the complex EC task interrelations of a known set of interactions e.g. Eckert at al. (Eckert *et al.*, 2009) and Wynn *et al.* (Wynn *et al.*, 2010), but the interactions remain within the product. Thus it appears as though the focus is identifying strategies for managing and executing changes effectively and efficiently, and not how changes within the given domain impact other domains (Hamraz *et al.*, 2013, Fricke *et al.*, 2000, Clark and Fujimoto, 1991).

Hamraz *et al.* propose that further research needs to be carried out on how ECs can be avoided through people-oriented measures e.g. better communication and knowledge sharing among designers and other disciplines (Hamraz *et al.*, 2013). However, they neglect to consider that ECs reflect the learning process of a given organisation as well as the need to adapt to new customer requirements. This view on the importance of ECs is taken by Wasmer *et al.* (Wasmer *et al.*, 2011) who indicate that ECs should actually be encouraged to produce a better and more reliable products and increase the productivity and efficiency of manufacturing systems. Thus, the gap identified by Hamraz *et al.* (Hamraz *et al.*, 2013) can be transformed to the question of how does communication and knowledge sharing enable more effective ECs, rather than to eliminate them. Furthermore, the complexity of products continues increase and as a result the impact that ECs have on the other domains are also more complex due to the number and type of interactions (ElMaraghy *et al.*, 2012). The impact of complexity on the other product realisation domains when attempting to execute ECs are increased lead times and costs. The lead times can be brought down if the EC is decomposed and tasks are, where possible, executed concurrently. This requires a high degree of collaboration between people and although there are numerous examples of the digitisation of the change management process in the literature (Huang *et al.*, 2001, Wasmer *et al.*, 2011), the use of digital domain models presents the opportunity to integrate ECM more readily with the engineering workflow. Therefore, the following section discusses the paradigm of MBSE and examples of how it has been used within the context of manufacturing.

2.3 *Tools and methods for manufacturing systems engineering*

As already alluded to in the introduction to this chapter, MBSE is a subset of MBE. Models are used extensively across a range of fields and disciplines as a way of representing a system for visualisation, testing, and validation. Within the context of MBSE a model is a digital representation of a system or entity. Such models can be generated through a number of software tools using languages that describe behaviour, geometry, relations etc. In order to effectively manage increasingly complex systems, the discipline of systems engineering has employed a number of lifecycle models. These lifecycle models are considered to have emerged from software development (Ruparelia, 2010, Estefan, 2007). It is important to note that the software development process can be quite different from systems engineering, of which manufacturing systems engineering is a subset. This is due to the fact that systems engineering, particularly manufacturing systems, have a myriad of perspectives such as electrical, electronics, mechanical, and of course software (Vogel-Heuser *et al.*, 2014). This means that the system development lifecycles that have been adopted are not necessarily fully appropriate for manufacturing systems engineering as they are ill-suited to managing the complexity of different modelling perspectives for a given system. There are three commonly cited lifecycle development models in the literature that are the Waterfall, V, and Spiral Lifecycle models and are illustrated in Figure 2-2, Figure 2-3 and Figure 2-4 respectively.

The Waterfall model (sometimes referred to as the cascade model) was originally proposed Benington (Benington, 1983) as a method for developing large computer programmes. The phases within this model underpin the lifecycle models that have proceeded it. The original model did not make considerations for unforeseen changes and modifications at the conclusion of each milestone. As a result, Royce extended the model with a feedback loop so that a preceding stage could be revisited should issues arise (Royce, 1987). Furthermore, identifying that there would be a need to revisit development phases that may not necessarily sit adjacent to each other, Royce added more complex feedback loops also that are represented as dashed arrows in Figure 2-2.

The V-model was developed by NASA (National Aeronautics and Space Administration) and first presented by Forsberg and Mooz (Forsberg and Mooz, 1991). The left leg of the V-model focuses on decomposing requirements while the right leg represents the solutions to address requirements and the process of integration, and verification. The more complex a system, the greater level of decomposition must occur. The V-model also has a z-axis that exists through the plane which accommodates multiple deliveries or multiple aspects of a given project.

Finally, the Spiral model was developed by Boehm (Boehm, 1988) that is based on the philosophy of “start small, think big.” The Spiral model addresses of the shortcomings of the Waterfall model in that it is incumbent on the development team to look-ahead so that reusability concerns are met as well as risks and issues that are often missed. While the Waterfall model is a specification driven approach, the Spiral model is considered to be risk-driven. The spiral moves through four phases as the development matures which are:

i. Determine objectives
ii. Evaluate alternatives and manage risks
iii. Develop and test
iv. Plan next iteration

Risk management in the Spiral model is used to assess the effort in terms of cost and time that is to be used for a given activity. Thus one of the key benefits of this lifecycle development model is the management of risks and costs at the outset. However, the Spiral model requires highly adaptive and agile project management as well as effective communication between domain stakeholders. Furthermore, it relies heavily on the ability to identify risks. Within complex system engineering environments, these shortcomings prevent the Spiral model from being employed effectively due to a lack of transparency and heavy administrative processes that are insufficiently reactive. As a result, despite the benefits of the Spiral approach, modern system engineering processes typically use a Waterfall model due to the clarity it provides and the fact that it aligns with existing management structures.

A hybrid of the Waterfall and Spiral approach was proposed by Iivari (Iivari, 1987) where provisions for baselines and milestones were made for each spiral cycle to facilitate control. While main phases would be risk-driven, sub-phases would be discipline driven. This was referred to as the hierarchical spiral model and provided a risk/cost conscious approach (Spiral) in conjunction with some level of domain stakeholder discipline (Waterfall).

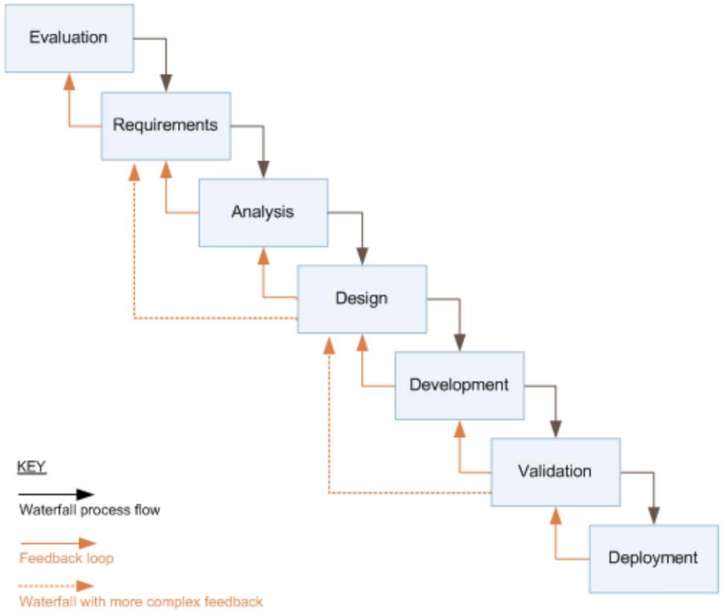


Figure 2-2 Waterfall model with Royce's iterative feedback (Ruparelia, 2010)

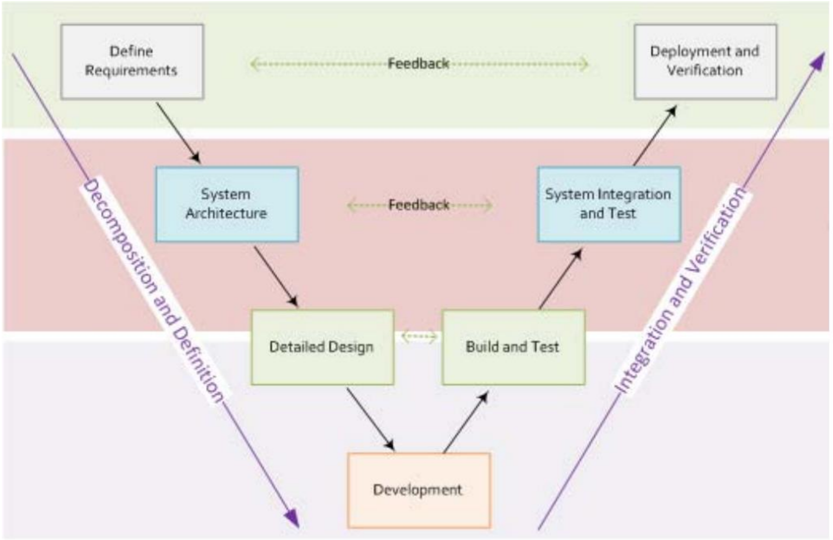

Define Requirements
Feedback
Deployment and Verification
System Architecture
Feedback
System Integration and Test
Detailed Design
Build and Test
Development
Decomposition and Definition
Integration and Verification

Figure 2-3 V-model using decomposition and feedback for verification and validation (Ruparelia, 2010)

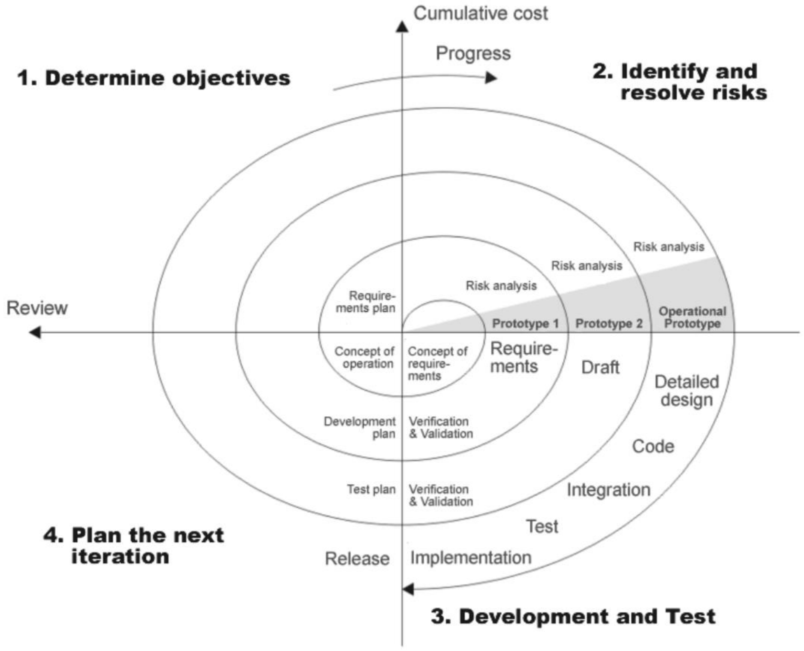


Figure 2-4 Boehm's spiral life-cycle (Ruparelia, 2010)

2.3.1 The use of MBSE for inconsistency management in automation systems

While lifecycle models allow stakeholders within management to track and control the development of a given system, there is still a need to connect output of a given phase or activity to the adjacent one. However, regardless of the decomposition, it is impossible to entirely decouple each activity or discipline. This overlap results in inconsistencies between the different domain models. The inconsistency is present when two or more statements are not jointly satisfiable (Spanoudakis and Zisman, 2001). This can commonly be attributed to human error, poor cross-disciplinary communication, and model complexity. Finkelstein is often credited with introducing the notion of inconsistency management (Finkelstein *et al.*, 1994). Building upon his work Spanoudakis and Zisman (Spanoudakis and Zisman, 2001) proposed a framework through which inconsistencies can be managed for software development: detection of overlaps, detection of inconsistencies, diagnosis, handling, tracking, and finally the application of an inconsistency management policy.

Gausemeier *et al.* propose a cross-domain base model at the conceptual design phase from which domain specific models emerge during the detailed design phase (Gausemeier *et al.*, 2009). A transformation engine propagates changes when they are made in a given domain, however this did not consider behavioural models. Hehenberger *et al.* (Hehenberger *et al.*,

2010) create a mechatronic ontology to check consistency which uses a Model/Analyser approach supported by rules. In Feldmann *et al.* (Feldmann *et al.*, 2015), Semantic Web Technologies (SWT) are used through the Resource Description Framework (RDF) where explicit links are formed between common concepts across heterogeneous models. SPARQL queries are then used to identify inconsistencies based on some user-defined bounds. Herzig *et al.* propose that a model can be represented by a graph and thus use an approach that uses graph pattern matching to check for inconsistencies (Herzig and Paredis, 2014). However, the approach is considered to be computationally expensive, although this has not validated through a case study.

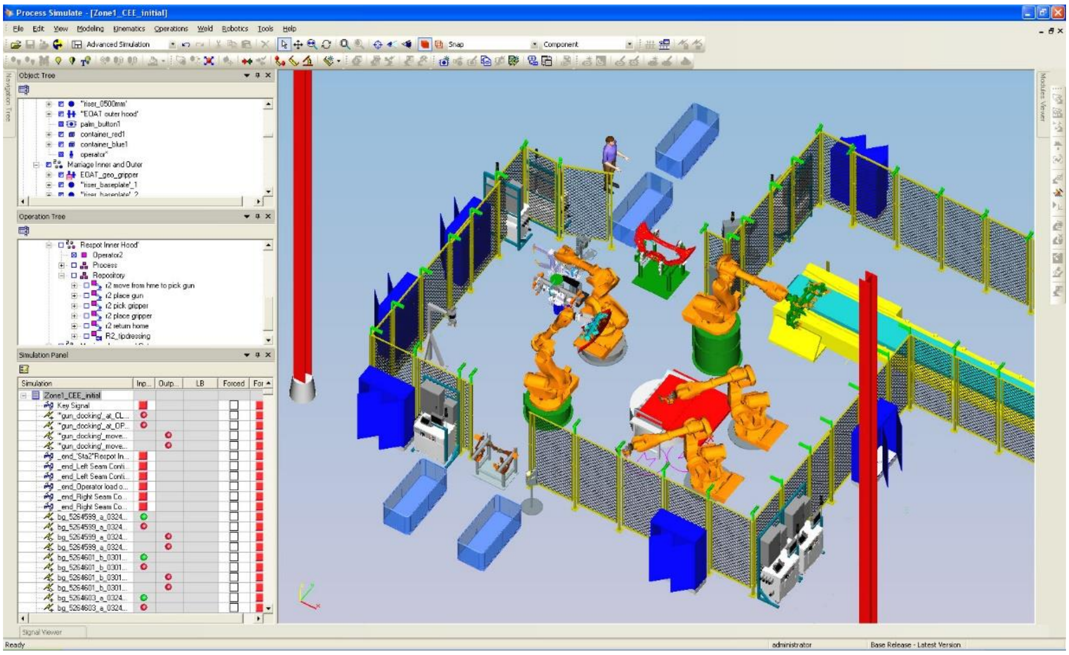
Outside of the industrial automation domain, Liu (Liu, 2013) addresses the problem of inconsistency in Unified Modelling Language (UML) diagrams. A total of 13 rules are presented which are checked through: manual checks, compulsory restrictions, automatic maintenance, and dynamic checks. This work focused on the design stage of the models and not how inconsistencies can arise during the use phase. Chavez *et al.* (Chavez *et al.*, 2016) state that the semantic gap that exists between certain languages is narrow i.e. C and Simulink, and therefore consistency can be checked more readily e.g. through Reactis[1]. However, where abstraction levels are different, inconsistencies are more difficult to identify due to the wider semantic gap. They describe an approach that checks consistency between UML and Java implementation called CCUJ (Conformance Checking between UML and Java) using constraints described using the Object Constraint Language (OCL). However, one of the limitations of the approach is the generation of false negatives i.e. inconsistencies would be detected even when there were none.

2.3.2 The Digitalisation of Manufacturing

Digital modelling and simulation solutions are used extensively in various engineering domains as they have the potential to enable the testing and validation of a system's behaviour and/or characteristics prior to physical implementation. This generally allows shorter and fewer design iterations and better design outputs (Harrison *et al.*, 2016). Figure 2-5 and Figure 2-6 are screenshots of the Siemens Process Simulate environment and Dassault Systemes Delmia environment respectively. These tools are focused on visualisation and validation of manufacturing systems with modules to support mechatronic devices, process industry systems, as well as provisions for capabilities such as virtual commissioning. These screenshots have been included to demonstrate that i)

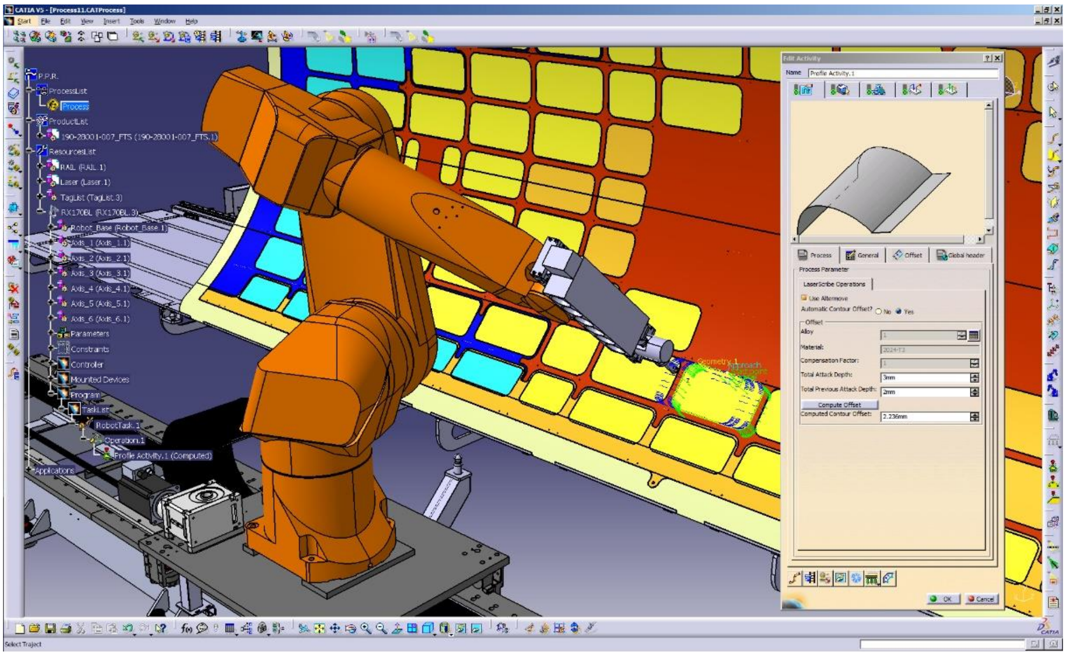
[1] http://www.reactive-systems.com

engineering models used are heavyweight as a consequence of representing complex geometries and details that may not necessarily be of value to the domain activity being executed, and ii) the differences in user interfaces highlighting the fact that software vendors are keen to leave their signature on their software. This, on one hand enables the user to navigate more easily within a broader package, but does hinder transition between the solutions of different vendors. Furthermore, the functionality of the respective software may be common however the semantics are often different (whether graphically or from a terminology perspective). This issue makes those that are experts in using a given engineering software to be averse to using another.



Simulation	Inp...	Outp...	LB	Forced	For
Zone1_CEE_initial					
Key Signal	🟥			□	🟥
"gun_docking'_at_CL...	🔴			□	🟥
"gun_docking'_at_OP...	🔴			□	🟥
"gun_docking'_move...		🔴		□	🟥
"gun_docking'_move...		🔴		□	🟥
end'Sta2"Respot In...	🟥			□	🟥
_end_Left Seam Conti...	🟥			□	🟥
_end_Left Seam Conti...	🟥			□	🟥
_end_Operator load o...	🟥			□	🟥
_end_Right Seam Co...	🟥			□	🟥
_end_Right Seam Co...	🟥			□	🟥
bg_5264599_a_0324...	🟢			□	🟥
bg_5264599_a_0324...	🔴			□	🟥
bg_5264599_a_0324...		🔴		□	🟥
bg_5264599_a_0324...		🔴		□	🟥
bg_5264601_b_0301...	🟢			□	🟥
bg_5264601_b_0301...	🔴			□	🟥
bg_5264601_b_0301...		🔴		□	🟥
bg_5264601_b_0301...		🔴		□	🟥
bg_5264603_a_0324...	🟢			□	🟥
bg_5264603_a_0324...	🔴			□	🟥

Figure 2-5 Screenshot of Siemens Process Simulate engineering environment



Name: Profile Activity.1

□ Use Altermove

Automatic Contour Offset? ○ No ● Yes

Offset

Alloy: 1

Material: 2024-T3

Compensation Factor: 1

Total Attack Depth: 3mm

Total Previous Attack Depth: 2mm

Computed Contour Offset: 2.236mm

Figure 2-6 Screenshot of Delmia engineering environment

Major CAD providers e.g. Siemens, have extended the capabilities of their respective engineering tools to more accurately model the PPR domain relations and, where possible, provided a mechanism for integrating the generated data within consistent modelling and simulation environments . The major benefit is to make unambiguous the implicit relationships that exist between domains, thus moving toward the objective of facilitating rapid, error-free change and re-design/-engineering of manufacturing systems in response to new requirements (e.g. product design, production process and volume changes). To provide value-adding engineering support i.e. design validation, optimisation, a variety of data types are required to implement executable models and simulations. However, the required sets of information are generally produced by various departments or organisations within the business or the supply chain are i) not available at the same point in time because of constraints inherent to the engineering process (Chandrasegaran *et al.*, 2013), and/or ii) likely to exists in a variety of data formats and standards (often digital but sometime not) depending on organisations, department, engineering domain, and software choices (Demoly *et al.*, 2013, Do, 2015).

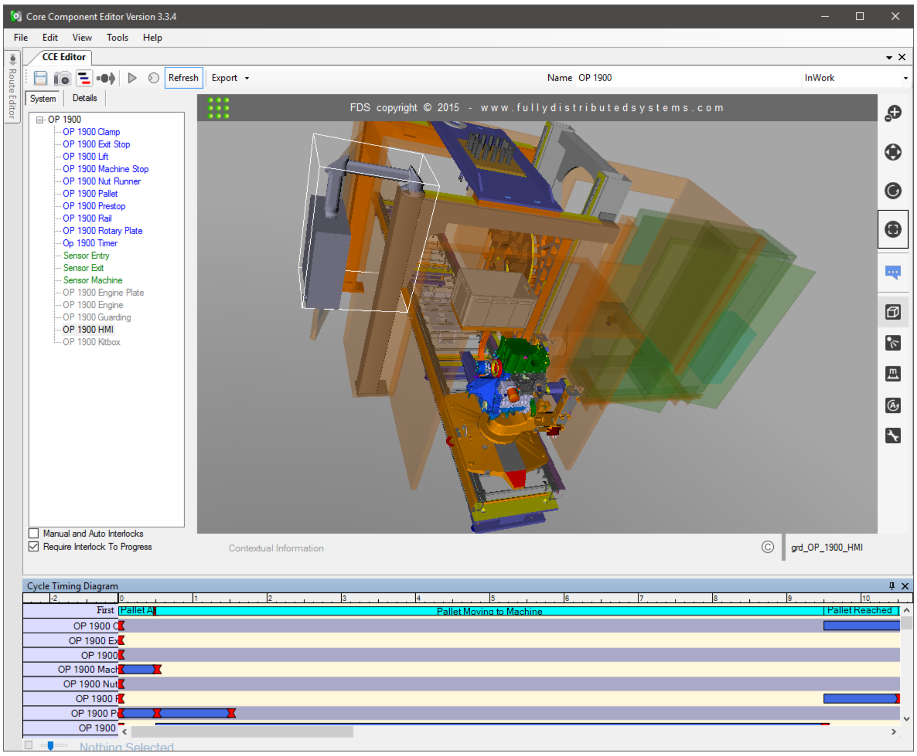
While CAD solution providers offer integrated solutions to gain control over data flows to align milestones (i.e. PLM/PPR central data base and data management systems) and potentially solve the problem of heterogeneous data formats by adopting proprietary data formats, their practical deployment is not always beneficial. Firstly, integrated solutions rely on several software modules to support modelling/simulation data editing and management, engineering workflows management, collaboration, etc. which are heavily reliant on so-called platforms e.g. Dassault 3DSexpericene . Secondly, the deployment of such solutions within an organisation requires re-defining, adapting, adding, and removing existing engineering processes. This is a time and resource intensive process which incurs a risk on existing projects occurring during the changeover phase. Finally, the costs (purchase, deployment, training, maintenance/support) associated with large scale integrated solutions can inhibit and often exclude supporting engineering organisations distributed across the supply chain. One of the few examples in the literature that systematically and critically discusses the shortcomings of commercial PLM offerings is (Hewett, 2009). Despite being a work published in 2009, many of the points raised are still valid today:

i) Lack of maturity of PLM solutions whereby a complex collection of tools is required which are then patched together in an ad-hoc way depending on the needs of the organisation

ii) Although engineers and designers are already on board with the software tools that they use, the implementation of PLM does not usually align with the practical workflows that they would prefer. Rather than applying a generic workflow to a given organisation, more could be done to include the respective stakeholders in the implementation process

Some of the shortcomings of PLM systems are addressed in this research work. The Evaluation chapter (chapter 5) continues the PLM discussion to compare and contrast the outcomes of this thesis.

Alternatives to major commercial solutions do exist, in the form of i) simpler platforms (e.g. Visual Components, vueOne) (Harrison *et al.*, 2016) , ii) ad-hoc integration of open source or low cost software modules to fulfil specific functions (Makris *et al.*, 2012, Exel *et al.*, 2014, Bergert and Kiefer, 2010, Kernschmidt and Vogel-Heuser, 2013, Vogel-Heuser *et al.*, 2014)) and iii) outsourcing of engineering services (e.g. Simulation Solution – SimSol , TCS Digital Solution) that make use of combination of the above mentioned software solutions to provide consultancy and engineering services related to Digital Manufacturing. In Figure 2-7 a screenshot of the core component editor within the vueOne engineering toolset is presented. It provides some similar functions to the tools illustrated in Figure 2-5 and Figure 2-6, but the simpler interface is a true reflection of its reduced capability as compared to commercial solutions. These tools are used and extended in this research with Chapter 3 describing the tool data model in more detail as well as the shortcomings addressed.



Name OP 1900

InWork

☐ Manual and Auto Interlocks
☑ Require Interlock To Progress

grd_OP_1900_HMI

Figure 2-7 Screenshot of the core component editor within the vueOne virtual engineering toolset

A number of academic groups have also presented digital manufacturing tools for virtual prototyping namely: Min *et al.* (Min *et al.*, 2002) who integrated real-time machine tool data within a virtual manufacturing environment. Suk-Hwan *et al.* (Suh *et al.*, 2003) developed Web-based virtual machine tools to interactively operate CNC machine tools, and Dietrich *et al.* (Dietrich *et al.*, 2002) presented sample scenarios for how the real and virtual processes could be integrated. While these tools show promise in their respective applications, they do not easily integrate with the other domain stakeholders. As a result, changes cannot easily be made unless these are discussed at the team layer which represents domain experts.

In summary, despite the benefits of digital manufacturing, the challenge of interoperability and data integration remains due to the breadth of design and engineering activities through the product realisation domains. There have been some examples in academia whereby there is a degree of integration across product realisation domains such as:

i. the generation of assembly sequences from product design CAD models in Pintzos *et al.* (Pintzos *et al.*, 2016) demonstrating a link between Product and Process domain models

ii. the reconfiguration of mobile robots based on process modifications in Angerer *et al.* (Angerer *et al.*, 2010) demonstrating a link between the Process and Resource domain models

However, such works have yet to move into industry as they are typically point solutions focusing on specific lifecycle phases with limited consideration of how they interact with other domains, disciplines, or phases. An understanding of domain activities coupled with knowledge-based models can provide a solution, regardless of the digital engineering tools used.

2.3.3 Towards integrating digital with physical

Classically, digital or virtual models are used to support in the design or engineering process but once the system has been commissioned such models are often never referred to later in the lifecycle. In fact, in the cases where these models are interrogated with a view to understand the layout, capability, or structure of a given system, such models are out-of-date and thus of little value. There remains a culture within industrial environments to store but not maintain models. This issue was scrutinised by a number of academics who wanted to explore what could be done with the heavy, complex, expensive engineering models created at the design and development phases of a manufacturing system.

Firstly, the notion of uni-directional digital data integration from the physical factory to the digital model was explored, resulting in the paradigm of the "Virtual Factory". This was with a view to enable more accurate simulations and as a result, more accurate predictions could be made from more relevant optimisation strategies based on real disturbance characteristics (Kuhn, 2006, Terkaj *et al.*, 2015). The Virtual Factory Framework (VFF) aimed to develop an integrated framework to implement the virtual factory (Sacco *et al.*, 2010, Tolio *et al.*, 2013). The project (funded in part by the European Commission) was self-described as *"An integrated virtual environment supporting the design and management of all the factory entities, ranging from the single product to the network of companies, along all the phases of the factory lifecycle"* (Sacco *et al.*, 2010). In addition to addressing the aforementioned lack of interoperability of engineering software, the project claimed to synchronise the real and the virtual factory. This synchronisation process was periodical and based on retrieving data from the real factory. The idea of the Virtual

Factory as per the descriptions in the references was to use historical data of the system and thus a real-time connection was not present.

However, Westkämper and Jendoubi had a more revolutionary vision when introducing the concept of the "Digital Factory", to support in the broader vision of the "Smart Factory" which proposed that data should flow bi-directionally between physical systems and digital models at real-time (Westkämper and Jendoubi, 2003). This enabled system monitoring, but in addition also allowed decisions made based on simulations models to be implemented, directly increasing system responsiveness and agility (Monostori *et al.*, 2016). At the time these ideas were conceived, computing power was at a premium, communication speeds were slower, and engineering software was insufficiently mature to cope with such requirements. Figure 2-8 illustrates the relationship between the Digital, Virtual, and Real factories. In summary, the Digital Factory integrates the Virtual Factory with the Real Factory. On its own, the Virtual Factory is able to simulate the Real Factory and support planning activities by importing data, while the Digital Factory enables a more permanent, operational connection between the respective factories.

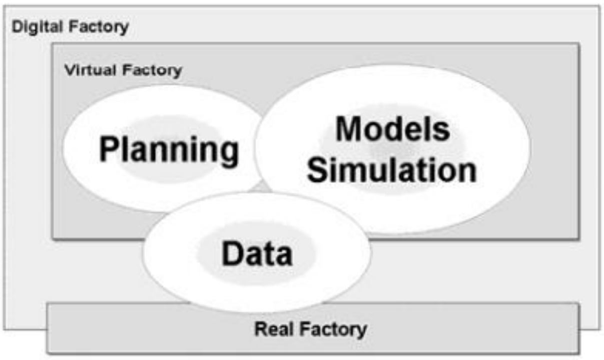


Figure 2-8 The relationship between the Digital Factory, Virtual Factory, and the Real Factory (Kuehn, 2006)

With the advent of increased computational power that can be encapsulated within smaller volumes and lower mass, high speed communication, and Internet connectivity i.e. IoT, there has been a paradigm shift towards Cyber Physical Production Systems (CPPS) that exploit the interaction of the cyber and the physical world (Monostori *et al.*, 2016). The CPPS consists of autonomous, cooperating elements that are to form connections dynamically to exchange information across all production levels. The aforementioned virtual models play a significant role to support the CPPS, but form part of a much broader system architecture.

Within the context of future manufacturing systems, one of the most cited architectures is RAMI 4.0 to realise Industry 4.0 (Figure 2-9) (Iarovyi *et al.*, 2016, Harrison *et al.*, Adolphs *et al.*, 2015, Hankel and Rexroth, 2015). Along the "Hierarchy Levels" axis the architecture complements the existing ISA-95 standard but extends it by adding the Product and Connected World layers, referencing a more intelligent product as well as a broader more connected set of enterprises that are able to exchange information. Along the "Life Cycle Value Stream" axis, the architecture respects the evolution of system entities, be they physical or cyber in nature, and their respective lifecycles. Finally, the "Layers" axis forms the most valuable part of the architecture to realise the vision of Industry 4.0. Along this axis, the transformation of assets from entities that generate data to agents that execute services at the business level is represented. In this thesis, the research aims to capture the knowledge that is generated through the **lifecycle**, from **products** all the way to the **manufacturing system**, with a view to transforming generic **information** to **functional** knowledge that can be exploited.

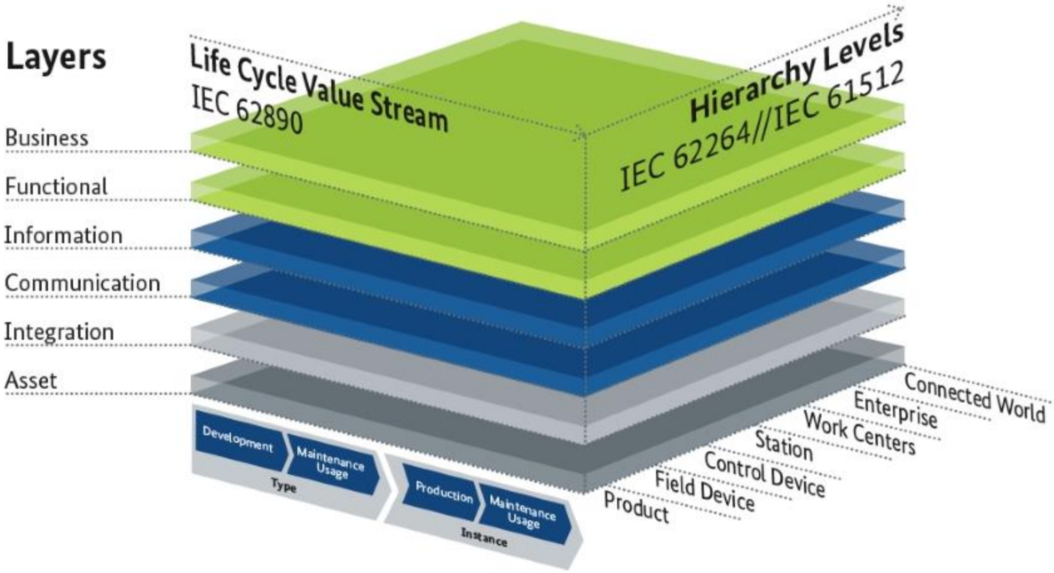


Figure 2-9 RAMI 4.0 (Hankel and Rexroth, 2015)

2.4 *Assembly process planning*

2.4.1 Background to Assembly Sequence Planning

One specific type of manufacturing process is assembly. This stage of manufacturing contributes up to 50% of total production time and accounts for more than 20% of total manufacturing cost (Rashid *et al.*, 2012). One of the key activities associated with realising an assembly is Assembly Process Planning (APP) (Jun *et al.*, 2005). In the literature, APP is decomposed into two further activities: Assembly Sequence Planning (ASP) and Assembly Line Balancing (ALB) (Rashid *et al.*, 2012). At a high level APP is focused on determining those set of processes that aggregate to realise an assembly within a given time. ASP is focused on converting the relationships that exist between product components into a sequential set of steps that are practically feasible. ALB is focused on ensuring that assembly stations have a balanced workload to prevent bottleneck or areas of starvation. In order to ascertain the amount of time it will take an activity to occur it is necessary to know the resources that are to be used, and to decompose the realisation of sequenced liaisons i.e. the output of ASP, into a more granularly defined set of tasks. There is therefore a close interaction between APP activities and manufacturing system design or reconfiguration (phase dependant).

Both ASP and ALB are referred to as NP hard problems (Rashid *et al.*, 2012). Within the context of ASP several methods have been developed to address this problem that can be classified into: graph/matrix-based, metaheuristics-based, and knowledge/artificial intelligence (AI) based (Chen *et al.*, 2010, Demoly *et al.*, 2011, Ahmad *et al.*, 2016). The graph-based approach is one of the earliest formal methods generating simple, undirected graphs to represent a product's topological structure, where nodes are components and edges are liaisons (Bourjault, 1986, De Fazio and Whitney, 1987), into directed graphs that show assembly direction based on constraints (Sanderson *et al.*, 1990). Based on the graphs, "cut-set" i.e. assembly by disassembly, methods were used to generate all possible assembly sequences, typically represented using AND/OR graphs (Homem de Mello and Sanderson, 1991). This method generated the complete set of assembly sequences which would become difficult to manage and represent as with complex products (Ben-Arieh and Kramer, 1994, Xu *et al.*, 1994). Matrices present the same information as graphs but in a more machine readable way, but both methods form the foundation of modern ASP methodologies.

The objective of metaheuristic approaches is to manage the large workspace associated with the ASP problem with complex products. Common methods include genetic algorithms (GA), ant colony optimisation (ACO), particle swarm optimisation (PSO), and simulated annealing (SA) (Rashid *et al.*, 2012, Wang *et al.*, 2009). These methods define a set of objectives for an algorithm e.g. minimum number of part orientations, with a view to deriving an optimum. These methods are used extensively and successfully in the ASP literature but they do suffer from tedious data entry processes, premature convergence, and high computational requirements (Rashid *et al.*, 2012, Wang *et al.*, 2012, Wang *et al.*, 2009).

Finally, there is the knowledge-based/AI approach to ASP. The tools used to facilitate these types of approach use new and novel models such as the connection-semantics-based-assembly trees (CSBATs) presented in Dong *et al.* (Dong *et al.*, 2007) that integrated geometry-based reasoning with knowledge-based reasoning to derive a sequence. PEGASUS (Product dEsign enGineering based on Assembly SeqUenceS Planning) was an assembly oriented design module for product lifecycle management (PLM) systems to facilitate concurrency across lifecycle phases i.e. product design and ASP (Demoly *et al.*, 2011). Design for Assembly (DFA) rules were captured mathematically and each assembly type (serial, parallel etc.) had a pre-determined cycle time to support process engineers in ascertaining the resources required. Kashkoush and Elmaraghy (Kashkoush and ElMaraghy, 2015, Kashkoush and ElMaraghy, 2014) described an approach that utilised pre-existing sequence knowledge to derive new sequences using a master assembly tree was proposed (see Figure 2-7). A similar approach that used pre-existing organisational knowledge concerning sequences was described in (Chen *et al.*, 2006) and applied to automotive body design. A knowledge-based approach using a three-stage optimisation method that culminated in a back-propagating neural network engine embedded within Siemens NX CAD tools was presented in (Hsu *et al.*, 2011). The major shortcoming of works that focus on ASP is the isolation of the method from practical workflows, particularly downstream to the Resource domain.

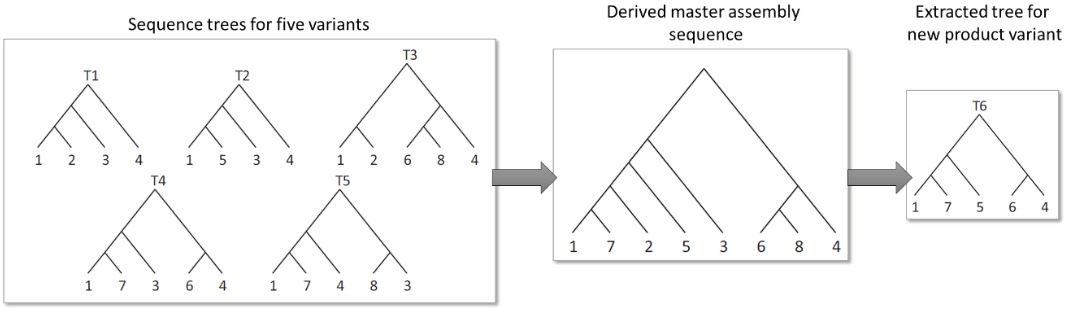

Sequence trees for five variants
T1
1 2 3 4
T2
1 5 3 4
T3
1 2 6 8 4
T4
1 7 3 6 4
T5
1 7 4 8 3
Derived master assembly sequence
1 7 2 5 3 6 8 4
Extracted tree for new product variant
T6
1 7 5 6 4

Figure 2-10 The use of pre-existing knowledge to generate a master sequence from which new sequence can be extracted. Adapted from (Kashkoush and ElMaraghy, 2015)

2.4.2 Connecting ASP with the Resource Domain

To address the shortcoming of literature that focuses solely on ASP, a number of works have extended knowledge-based approaches to assimilate information across domains and phases.

The work of Yang *et al.* (Yang *et al.*, 2016) proposed a four-layer framework for manufacturing process information based on a metamodel (see Figure 2-11). The major focus of the work was maintaining consistency, accuracy, completeness, and generality between process planning activities and the manufacturing system. One of the key insights of the work was recognising the need for a layered framework that considered abstracted concepts (in the metamodel) down to instantiation of specific data in the data layer. This layered approach has been adapted and extended in this thesis. One of the shortcomings of the work by Yang *et al.* (Yang *et al.*, 2016) was its high level implementation using UML and thus an inability to exploit the semantics proposed.

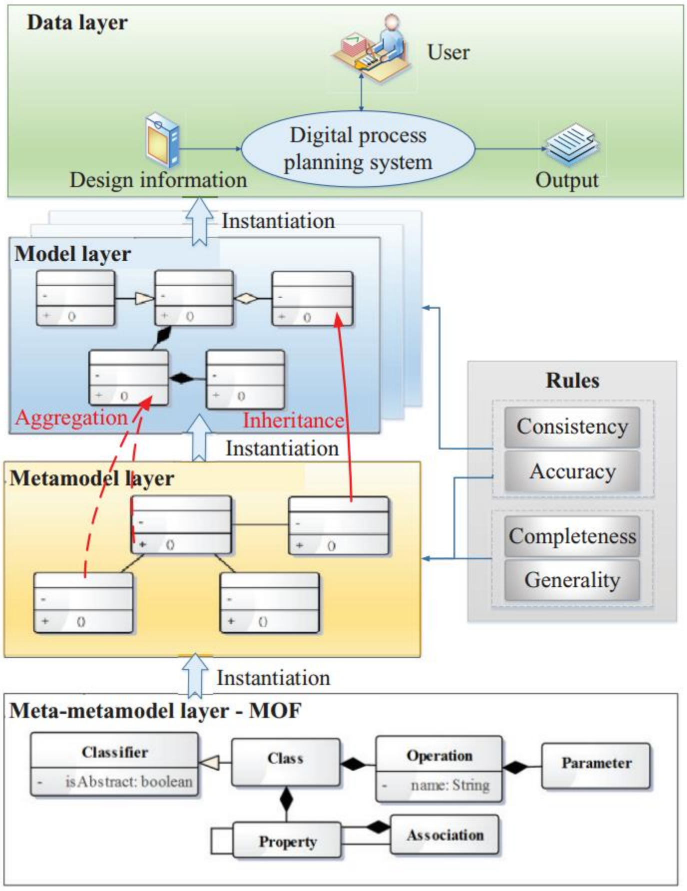


Figure 2-11 Framework for manufacturing process information modelling proposed by (Yang et al., 2016)

In Zha *et al.* (Zha *et al.*, 1999) a concurrent product design and assembly planning (CDAPFAES) methodology was proposed that included conceptual design, detailed design, assemblability analysis, DFA, assembly system design, APP, simulation, and techno-economic analysis. The approach was further developed in Zha *et al.* (Zha *et al.*, 2001b) and then implemented in Zha *et al.* (Zha *et al.*, 2001a) where it was renamed to Assembly-Oriented Design Expert System (AODES). The system allowed the given stakeholder i.e. product designer, to generate, modify, and analyse a product through its design stages i.e. concept design, detailed design etc. In order to capture designer knowledge and requirements, interaction between human and computer was enabled through questions to provide decision support. In Su and Smith (Su and Smith, 2003) a method that integrated DFA, APP, and production simulation was presented. An integrated framework, Assembly-Oriented Product Design and Optimisation (AOPDO), used a function modelling approach to structure functions, activities, and processes for a given system model be it within the Product domain or the Resource domain using IDEF0 (Mayer, 1992). Details of the approach were not fully elaborated and production simulation was limited to petri nets so there was a lack of virtual modelling. The Sequence Planning and Design Environment (SPADE) method presented both a hierarchy of the product model and for component liaisons that enabled the mapping of assembly actions (Barnes *et al.*, 2004). Although this work did not fully extend into identifying the resources that would be used to execute said actions, embedding assembly actions within the methodology provided a link to the Resource domain.

An early example of an attempt to integrate process planning with machine software is presented in Feng and Song (Feng and Song, 2003). The work culminated in the development of Part 2 of ISO 16100 which focused on providing a standard view of information models for interoperability in industrial automation systems. The work used UML to represent the process information and considered a broad range of manufacturing and assembly processes as well as categorising manufacturing system equipment and the parameters associated with it. Figure 2-12a and Figure 2-12b are extracts of the standard illustrating the assembly process and manufacturing system resource models respectively. The reader should note the similarity in structure, terminology, and properties of the respective classes as compared to the ontological models that are presented later in this chapter. Although not referred to as an ontology in the standard itself, the work does align with the definition. The major shortcoming is its representation within UML and no implementation within a language that supports knowledge representation e.g. RDF/OWL. Another shortcoming of the model is the lack of the link to machine control and thus preventing explicit mapping between the process model and the resource model.

Despite its comprehensive, structured approach to capturing process and manufacturing information, the standard has seen limited use in both industry and academia, evidenced by the lack of its use in the literature. This may be because it does not address the complex semantic relations between the concepts described, a resistance from software vendors to embrace open standards, or simply a lack of publicity concerning the standard.

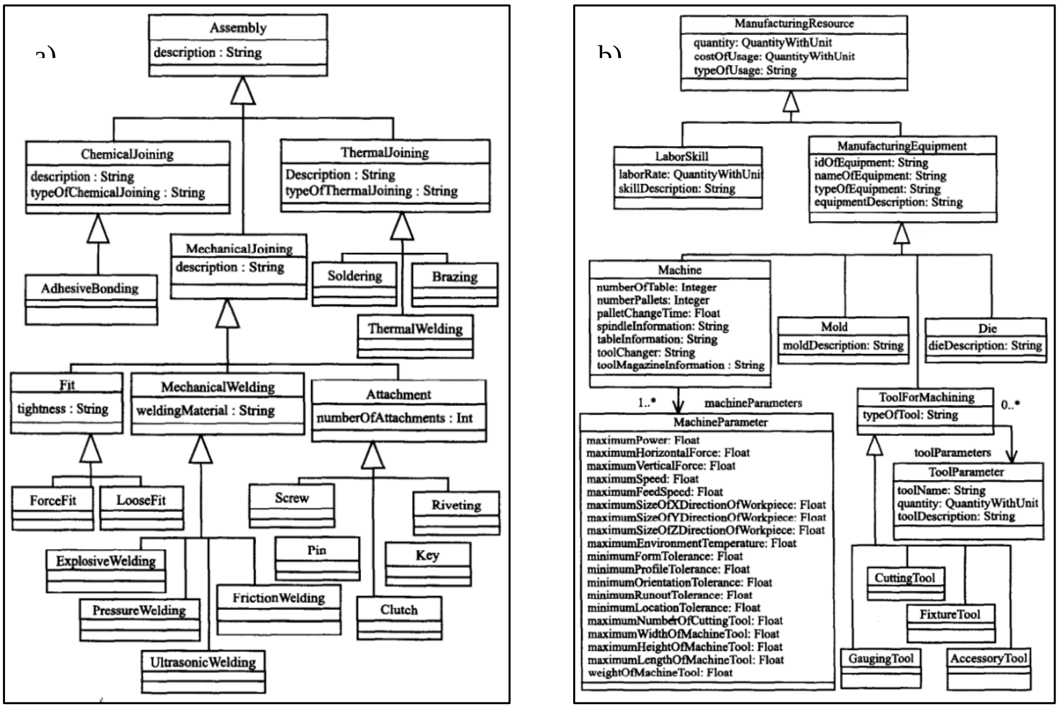


Figure 2-12a) Process information model and b) manufacturing process model from (Feng and Song, 2003)

The use of function blocks to support in adaptive APP is described in Wang *et al* (Wang *et al.*, 2008, Wang *et al.*, 2012). For a given assembly sequence, assembly features are mapped to assembly feature function blocks. An assembly feature is defined as the connection between two mating components. Figure 2-13 illustrates an example of implementation of the adaptive function blocks proposed in Wang *et al* (Wang *et al.*, 2008). The management function block (M-FB) is an execution manager that handles what cannot be managed by assembly feature function blocks (AF-FB). After every assembly task is fulfilled the proceeding AF-FB is called based on the assembly sequence. One of the key benefits of the work was proposed to be the reusability of function blocks as they encapsulated knowledge about how an assembly task should be executed within an algorithm. However, the work did not describe how the sequence would be checked for consistency when the function block has been instantiated. Furthermore, the issues associated with semantics are not addressed i.e. different stakeholders, industries, or product designers will name assembly features in a different way.

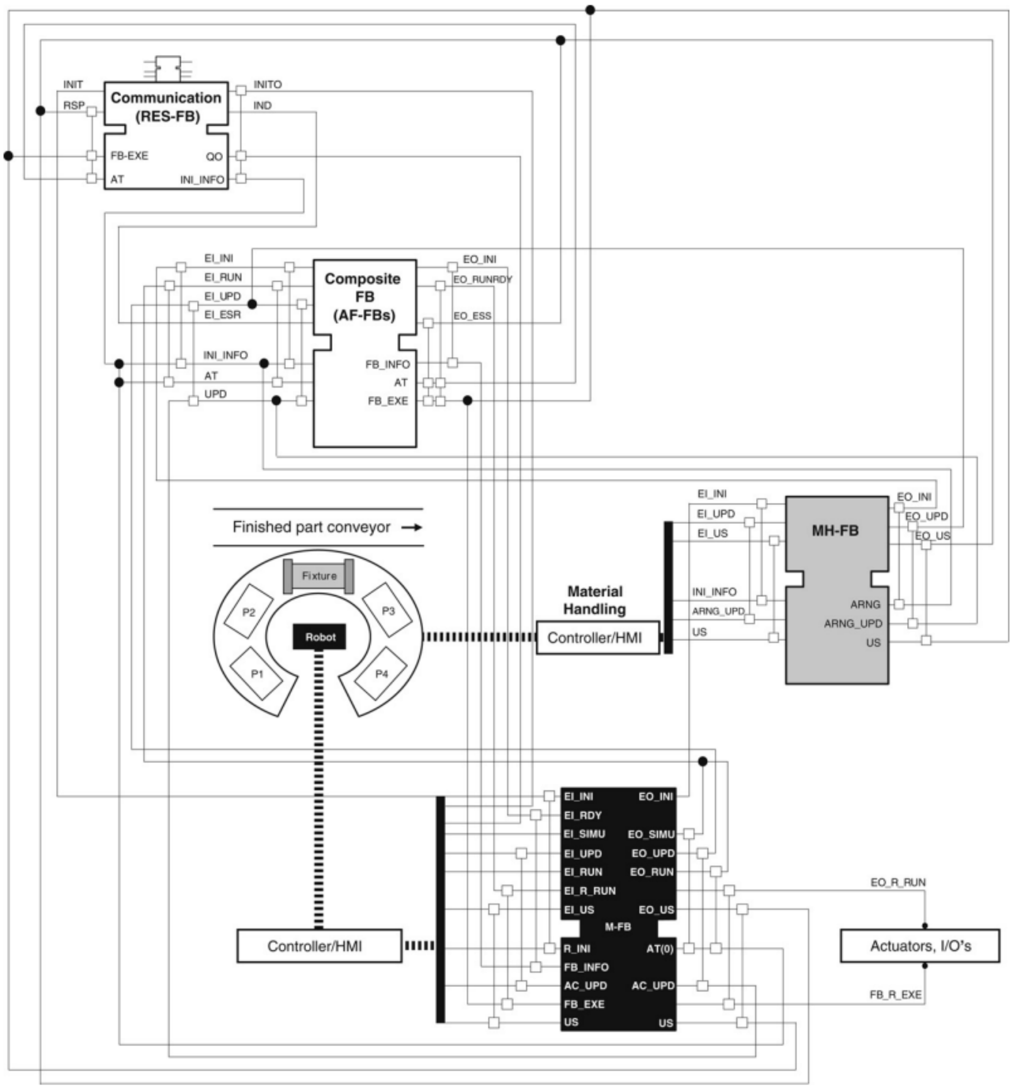


Figure 2-13 Example of function block in an assembly cell FB network from (Wang et al., 2008)

In Proctor *et al.* (Proctor *et al.*, 2016) the use of Product and Manufacturing Information (PMI) is used to automate robot planning in conjunction with the Robot Operating System (ROS). Due to the recent development of semantic PMI in ISO 10303 AP 242, geometric dimensioning and tolerance (GD&T) requirements are carried through from design to robot process planning without human intervention. The work of Michniewicz (Michniewicz and Reinhart, 2015, Michniewicz and Reinhart, 2014, Michniewicz *et al.*, 2016) presented a framework that spanned ASP through to machine control code, again with a primary focus on robotic assembly cells. The method automatically derives an assembly sequence from CAD drawings through an "assembly by disassembly approach" that is supported through some manual efforts should any preference or pre-existing knowledge need to drive the final solution. The sequence is transformed into a set of process primitives (similar to Barnes *et al.* (Barnes *et al.*, 2004)) and then, based on a skill model, the capabilities of a robot transform process primitives into control code. Due its focus on robotic cells, the method needs to be proved on more bespoke and thus complex equipment, and furthermore it is not clear how this would link with IEC 61131-3 (Hanssen, 2015). An overview of the methodology described by Michniewicz is illustrated in Figure 2-14.

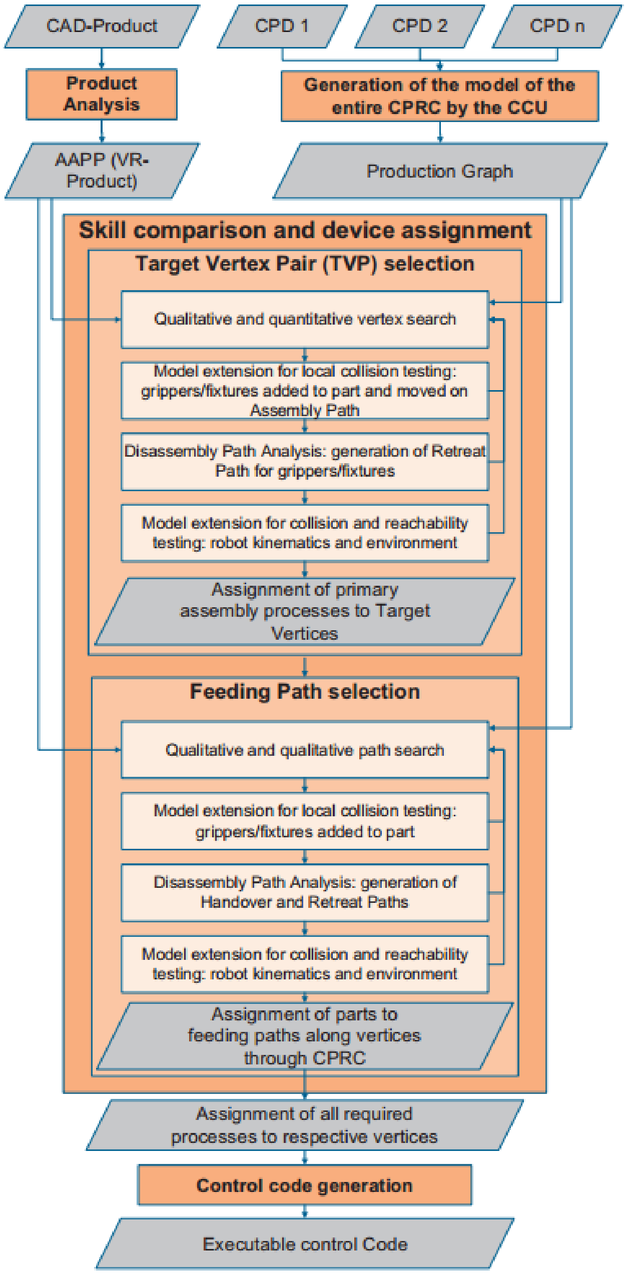

CAD-Product
CPD 1
CPD 2
CPD n
Product Analysis
Generation of the model of the entire CPRC by the CCU
AAPP (VR-Product)
Production Graph
Skill comparison and device assignment
Target Vertex Pair (TVP) selection
Qualitative and quantitative vertex search
Model extension for local collision testing: grippers/fixtures added to part and moved on Assembly Path
Disassembly Path Analysis: generation of Retreat Path for grippers/fixtures
Model extension for collision and reachability testing: robot kinematics and environment
Assignment of primary assembly processes to Target Vertices
Feeding Path selection
Qualitative and qualitative path search
Model extension for local collision testing: grippers/fixtures added to part
Disassembly Path Analysis: generation of Handover and Retreat Paths
Model extension for collision and reachability testing: robot kinematics and environment
Assignment of parts to feeding paths along vertices through CPRC
Assignment of all required processes to respective vertices
Control code generation
Executable control Code

Figure 2-14 Overview of methodology for connecting APP with machine control code (Michniewicz and Reinhart, 2015)

On the other hand a heavy focus on supporting changes to PLC code is presented by Lennarton and Bengtsson (Lennartson *et al.*, 2010, Bengtsson *et al.*, 2013, Bengtsson and Lennartson, 2014). This set of work addresses the lack of flexibility of existing PLC code, attributing it to the lack of decoupling between core concerns which are related to operations defining the behaviour of the system and support concerns such as alarms, safety, manual control etc. To address this problem an aspect-oriented programming approach is used that culminates in the development of a graphical language called sequence of operations (SOP). There are similarities between SOP and sequential function chart (SFC), the differences are largely semantics. This language is used in a prototype tool called Sequence Planner. The first part of the work that sets the groundwork for the language decomposes the problem into PPR domains and relations are discussed in much the same way as PPR ontologies (Lennartson *et al.*, 2010). The integration of the approach with PLM tools is not discussed and so visualisation through virtual models for validation is not addressed, although this is mentioned to be a part of future work. Furthermore, as the approach was not implemented using ontologies, the ability to infer the impact of change and make modifications to the sequence is not possible.

2.5 *Ontologies and Knowledge Representation*

Artificial Intelligence (AI) is the study of intelligence with a view to replicating and implementing it on computer systems [32]. Knowledge Representation (KR) is a form of AI that focuses on modelling concepts which are both human and machine readable utilising semantics for system description. This allows questions to be asked and answered on the basis of which decisions can be made, either using computer-based algorithms or through human experts. The aforementioned PLM tools use a very basic form of KR, but full realisation is hindered by the problems described in the previous section i.e. proprietary formats. KR is envisioned to move beyond current approaches for generating, integrating and managing data, such that true concurrent and collaborative engineering can be realised.

A key benefit of KR is in its potential to automate processes, with this research focusing on the engineering workflow. Knowledge can be modelled, mapped, and linked using different methods such as: ontologies, Linked Data, and rules or frames (Brachman, 2004). In KR, a Knowledge Base stores the knowledge model, which can be accessed to be queried and/or updated.

The word “ontology” has a different meaning depending on the context. Firstly, there is the philosophical discipline which is an uncountable noun written as “Ontology” which deals with nature and the structure of “reality” (Guarino *et al.,* 2009). Aristotle dealt with this subject and defined Ontology as the “science of being”. Unlike the scientific ontology, this branch of metaphysics focuses on the nature and structure or reality independent of how this information would be used.

On the other hand, the use of ontology in this research stems from the field of Computer Science whereby it refers to a type of information object. An ontology is a form of KR and defined by Gruber (Gruber, 1993) as “an explicit specification of a conceptualisation” while Borst (Borst *et al.*, 1997) extends this definition to “shared conceptualisation”. Ontologies are a form of knowledge representation for a given domain through the use of formal semantics and can be used to arrange and define concept hierarchy, taxonomy, and topology. Ontologies differ from a database approach as their focus is the preservation of meaning to facilitate interoperability, while the main purpose of database schema is to store and query large data sets (Martinez-Cruz *et al.*, 2012). Ontologies can be accessed for querying and/or modification purposes and they can be implemented using several semantic languages (Kalibatiene and Vasilecas, 2011). Resource Description Framework (RDF) based languages remain dominant which are based on XML, part of the World Wide Web Consortium (W3C) recommendations . RDF-based models (i.e., RDF graphs) are set of triples composed of a subject, a predicate, and an object. This structure to information description mimics natural language. The Web Ontology Language (OWL) (McGuinness and Van Harmelen, 2004) is an enriched extension of RDF that has the capability to model cardinality constraints, enumeration, and axioms resulting in a richer more accurate model. Figure 2-15 illustrates the language architecture described.

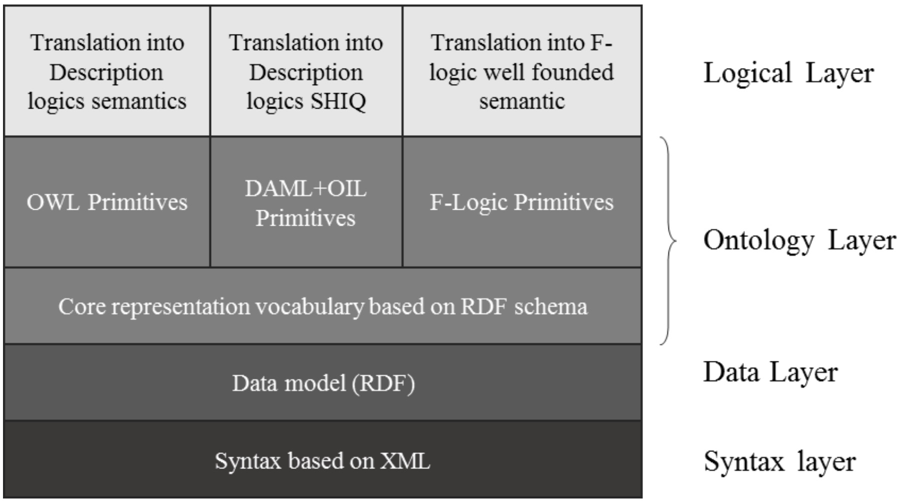

Translation into Description logics semantics
Translation into Description logics SHIQ
Translation into F-logic well founded semantic
Logical Layer
OWL Primitives
DAML+OIL Primitives
F-Logic Primitives
Ontology Layer
Core representation vocabulary based on RDF schema
Data model (RDF)
Data Layer
Syntax based on XML
Syntax layer

Figure 2-15 Representation layers for ontologies adapted from Lin et al. (Lin et al., 2004)

The information from OWL models can be queried using RDF-based query language (SPARQL) (Prud’Hommeaux and Seaborne, 2008). In addition, SPARQL update (Seaborne *et al.*, 2008) can be used for retrieving and updating ontological models. Rule-based languages such as the Semantic Web Rule Language (SWRL) (Horrocks *et al.*, 2004) can be employed within ontologies. These rules are defined on top of such ontological models, as presented in (Puttonen *et al.*, 2013). Through the use of rules and RDF triples, semantic reasoning engines can infer implicit knowledge and validate the consistency of a model.

2.5.1 Types of ontologies

In Usman (Usman, 2012) a classification of ontologies within the context of manufacturing systems engineering is presented. The two criteria are the level of formalisation and the level of specificity. In the former, there exist Lightweight and Heavyweight ontologies, while in the latter there exist Foundational, Core, and Domain ontologies.

2.5.1.1 Levels of ontological formalisation

Lightweight ontologies are based on simple taxonomies with simple parent child relationships between concepts (Borgo and Leitão, 2007). Examples of these types of ontologies are WordNet (Miller, 1995), as well as a number of international standards within the context of product data management e.g. STEP (ISO, 2011). These types of ontologies have limited concept constraints such that their semantics are insufficient to support interoperability i.e. to integrate different domain models (Dartigues *et al.*, 2007). To address this, particularly for the STEP format, the ONTOSTEP ontology was developed which addressed the lack of logical formalism of EXPRESS so that reasoning and semantic operability could be realised (Krima *et al.*, 2009). This brings the advantage of inference capabilities and thus allows them to address interoperability issues.

2.5.1.2 Levels of ontological specification

Foundational ontologies aim to cover the semantics of "everything" and thus cover the semantic base for any given domain. Examples of foundational ontologies include DOLCE (Masolo *et al.*, 2003) and the Basic Formal Ontology (BFO) (Smith and Grenon, 2002). The concepts in Foundational ontologies are generic and as a result are often too broad to be used in a practical engineering context.

Core ontologies are limited in the literature and sit at a level of specificity between Foundational and Domain ontologies. The objective of Core ontologies is to cover a set of semantics that are shared across multiple domains (Deshayes *et al.*, 2007). As a result, they lend themselves to reuse and are of particular importance within the context of interoperability. As with ontologies more generally, the shortcoming of Core ontologies remains the lack of “shared conceptualisation” between practitioners and developers. Focusing on the semantics i.e. the set of generic concepts that should exist within a given ontology is insufficient to encourage the application of Core ontologies.

The author takes the stance that the most effective realisation of Core ontologies (and other ontologies more generally) is in part the terminology used to describe the system in question, but also to enrich said terminology with meaning derived from ***relations*** with other concepts. This formation of graph patterns can in turn be analysed and inferences made that enable the identification of a common entity under multiple aliases.

Finally, Domain ontologies have the greatest level of specificity and due to their focus and distinct semantics, interoperability between Domain ontologies is challenging. Within the context of supporting manufacturing system lifecycles, it is therefore incumbent on the Domain ontology development team to identify Domain touchpoints and ensure that links and mappings exist between the relevant concepts.

2.5.2 PPR Modelling

There is a rapidly growing body of literature in the area of PPR modelling. This section focuses on the methods used to create these models, how and where they have been used and the respective shortcomings to highlight both the technical and knowledge gaps.

Rampersad (Rampersad, 1994) introduced an integrated PPR model that integrated the PPR domains in 1994. He decomposed the domains and showed how and where different areas should be linked. A key insight of Rampersad was to link the product domain to the assembly system as well as to the process resulting in the formation of the integrated assembly model illustrated in Figure 2-16. This was with a view to realise concurrent engineering. However, at the time of publishing, there was a lack of computing power, engineering tools and industrial consensus on the approach, therefore this remained a conceptual work.

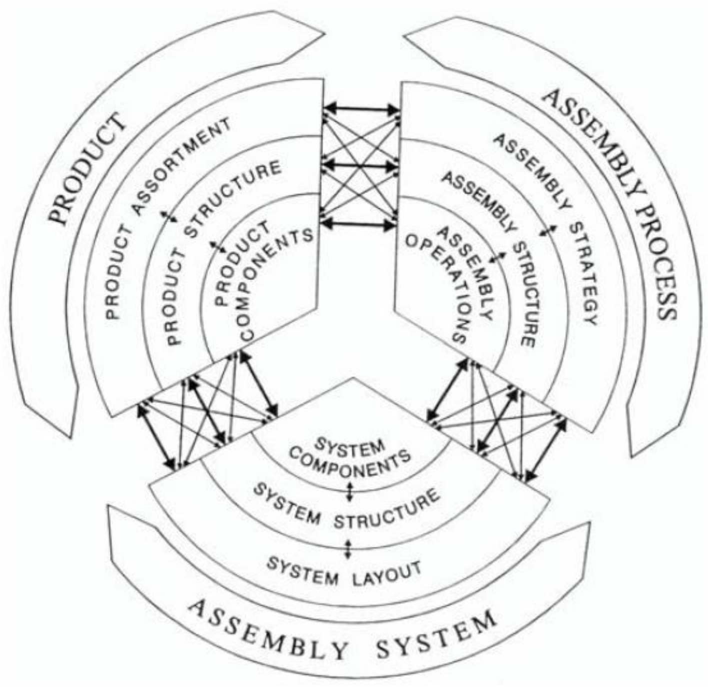


Figure 2-16 Integrated assembly model (Rampersad, 1994)

Delamer *et al.* (Delamer and Lastra, 2006) described how ontological models supported by semantic web services could be utilised to rapidly reconfigure manufacturing systems. However, the work did not embed the ontology with an ability to recognise functional aspects of processes and equipment, preventing automatic selection and invocation of manufacturing processes. Further, there was limited description of the product ontology and no explicit identification of how the respective ontologies were linked.

Lohse (Lohse, 2006) on the other hand did provide insight with regards to the mappings within and across domains using a function-behaviour-structure framework in the ONTOMAS framework (Figure 2-17). However, Lohse's approach to domain integration detracted from Rampersad's in that it utilised the Process domain as the "middle man" i.e. there was no explicit link between the Product and Resource domain (although a "port" concept did permit the representation of interrelations between the Product domain and the Resource domain (Lohse *et al.*, 2005, Lohse *et al.*, 2004)) This had the consequence of the inability to query system suitability with respect to product and vice versa, preventing the full exploitation of the presented ontological models.

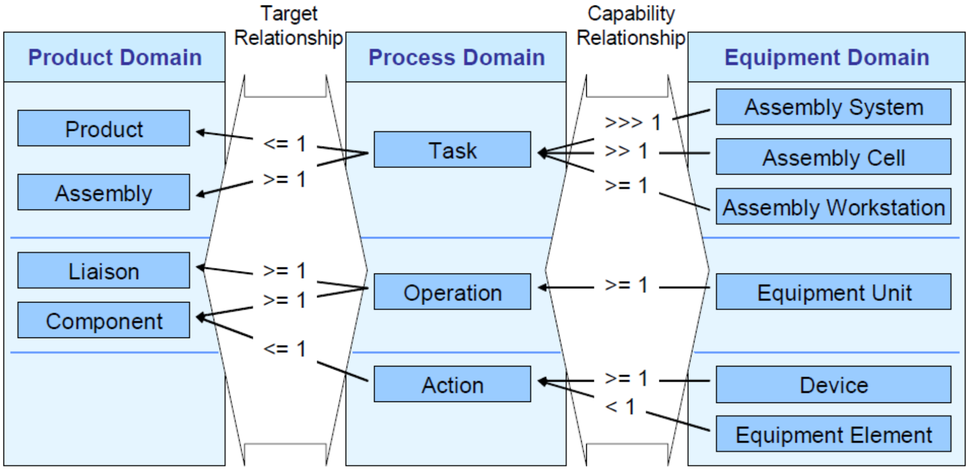


Figure 2-17 High level view of the ONTOMAS ontology describing inter-domain links (Lohse, 2006)

The work of Lanz (Lanz, 2010) aligns well with the work presented in this research paper as it addressed the same problems: i) to give meaning to the large amount of data and information that exist in organisations, and ii) to have decision support systems that can be trusted by designers and engineers. Lanz therefore created a PPR ontology and showed how the respective domains are linked (Figure 2-18). The work demonstrated how data from engineering tools could be imported into the ontology, although the reverse was not described. In addition, there was limited description as to how the data extracted and linked from the engineering tools can be exploited and manipulated.

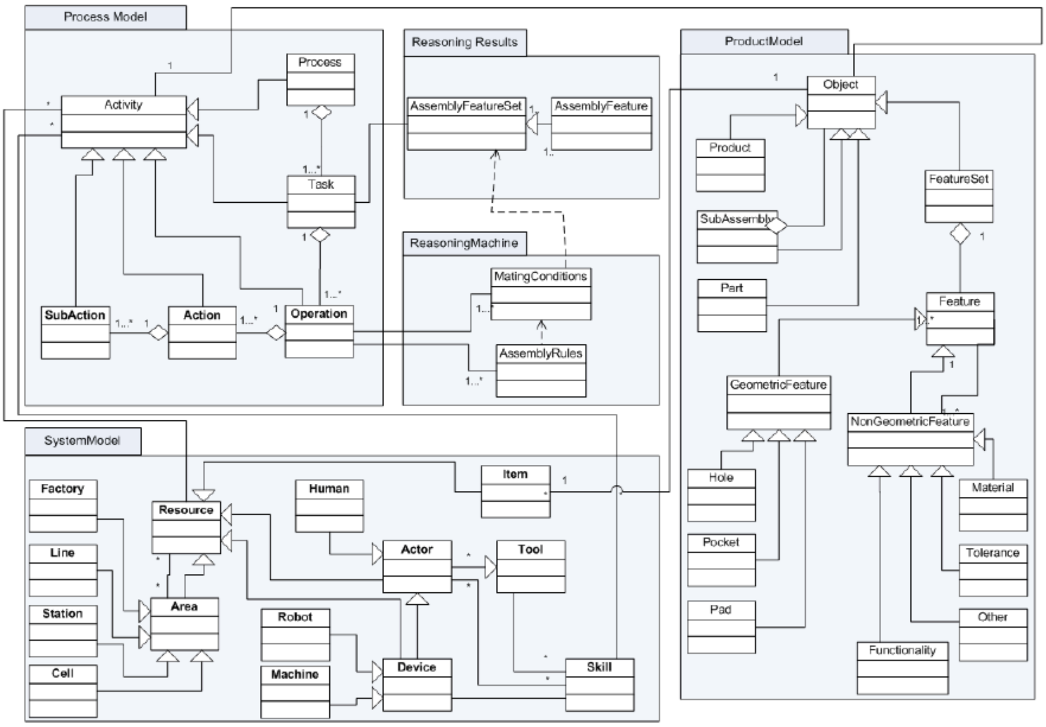


Figure 2-18 Product-Process-System Model (Lanz, 2010)

The concept of “feature” was used by (Hasan *et al.*, 2014) to carry information concerning a product through process planning and the engineering of shop floor devices such as grippers to map to skills. Hasan *et al.* then extended this work in Hasan *et al.* (Hasan *et al.*, 2016b, Hasan *et al.*, 2016a, Hasan and Wikander, 2017 , Hasan and Wikander, 2016) to directly extract product assembly feature data from SolidWorks through an application programming interface (API) using a boundary representation methodology.

MASON (MAnufacturing Semantics ONtology) was proposed by Lemaignan *et al.* which demonstrated automated cost estimation and semantic-aware multi-agent system for manufacturing (Lemaignan *et al.*, 2006). However, being self-described as an upper-ontology, it was too abstract to be used as a practical engineering tool and is similar in scope to the work of Ahmad *et al.* (Ahmad *et al.*, 2015a). The main concepts and object properties of the MASON ontology are illustrated in Figure 2-19.

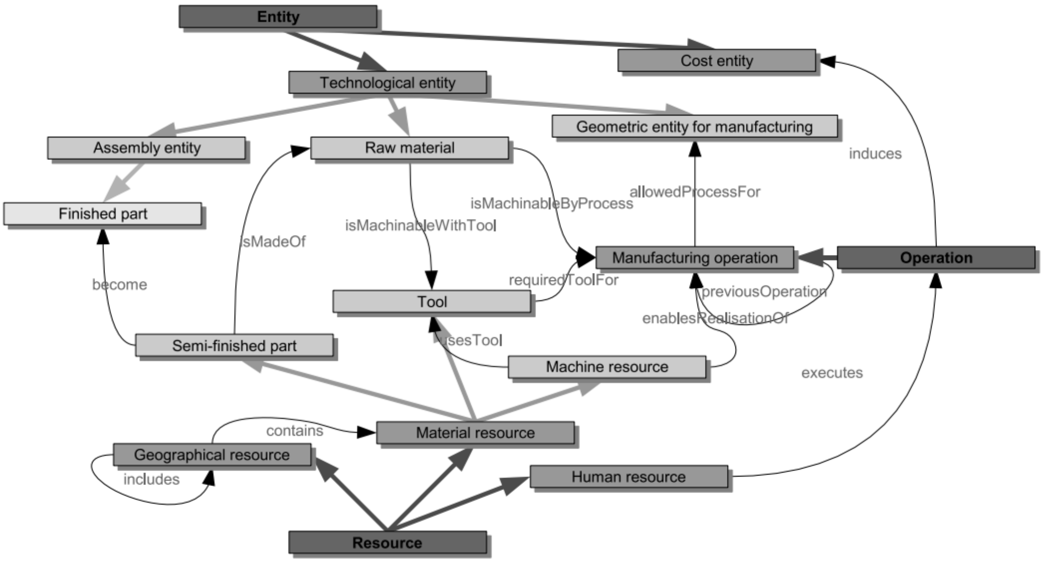


Figure 2-19 Main classes and object properties of the MASON ontology (Lemaignan et al., 2006)

Panetto *et al.* present another manufacturing related ontology called ONTOPDM (ONTOlogy for Product Data Management) which is an approach for facilitating system interoperability within manufacturing environments (Panetto *et al.*, 2012). To manage heterogeneous data sets, Panetto *et al.* utilised existing standards for product technical data (IEC 10303) and Enterprise Resource Planning/Manufacturing Execution System data (ISO 62264). However, the research focused primarily on the Product Domain and how information could be exchanged with reduced semantic uncertainty. As a result the other product realisation domains were less well defined. In addition, by creating an ontology that linked to a set of standards, Panetto *et al.* concluded that it was necessary to further extend the ontology to include other standardisation initiatives. This poses the risk of a large, monolithic ontology that may be difficult to maintain.

A manufacturing systems engineering (MSE) ontology was presented by Lin and Shahbaz in (Lin *et al.*, 2004). The work formed part of a broader, extended enterprise system called the EEMSE moderator (the acronym was not elaborated in the work). A moderator was defined in the work as “an intelligent support application designed to facilitate and improve concurrent engineering design by enhancing the degree of awareness, cooperation, and coordination among engineering team members”. The research noted two key issues when considering a multi-enterprise, complex, engineering projects. Firstly, design change information needs to be effectively communicated to relevant stakeholders and secondly what is perceived to be **important** aspects of a given design by a given stakeholder need to be expressed explicitly. The work culminated in the development of a large, complex ontology with the top level abstract classes illustrated in Figure 2-20. This monolithic, non-modularised model hindered reuse and thus the authors of that paper do not have appeared to extended the work as discussed in the concluding section of the paper which was to support more powerful query and inference.

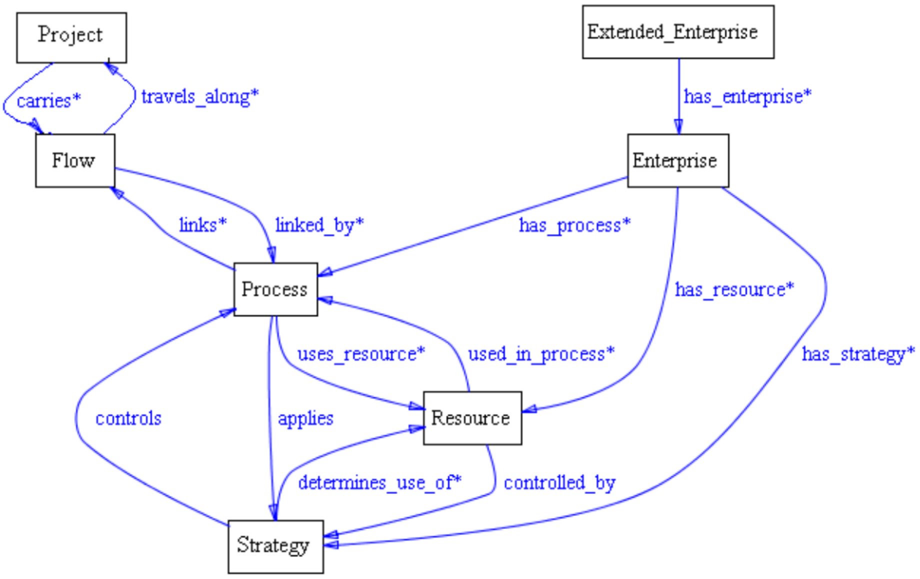


Figure 2-20 Top-level abstract classes from the MSE ontology (Lin et al., 2004)

Recently, Chhim *et al.* (Chhim *et al.*, 2017) presented a product design and manufacturing process based ontology to support manufacturing knowledge reuse. They identified that a lack of ***granular*** mappings between product design (the Product Domain) and the manufacturing process (Process Domain) did not exist, and hypothesised that this was one of the reasons for the lack of industrial uptake of ontologies. The authors focused on reusing the knowledge generated from DFMEA (design failure modes and effects analysis) and PFMEA (process failure modes and effects analysis) processes. The full design and manufacturing ontology is presented in Figure 2-21 and it was used to identify how a given product component could fail and what detection controls had been used in the past using a SPARQL query. Although the research identified the lack of granular mapping in existing works, examination of Figure 2-21 reveals a lack of mapping at the lowest levels of DFMEA (left branch) and PFMEA (right branch). In order to identify relationships, complex queries would need to be written. However, this could be addressed through SWRL rules that could automate low level mappings and thus simplifying queries.

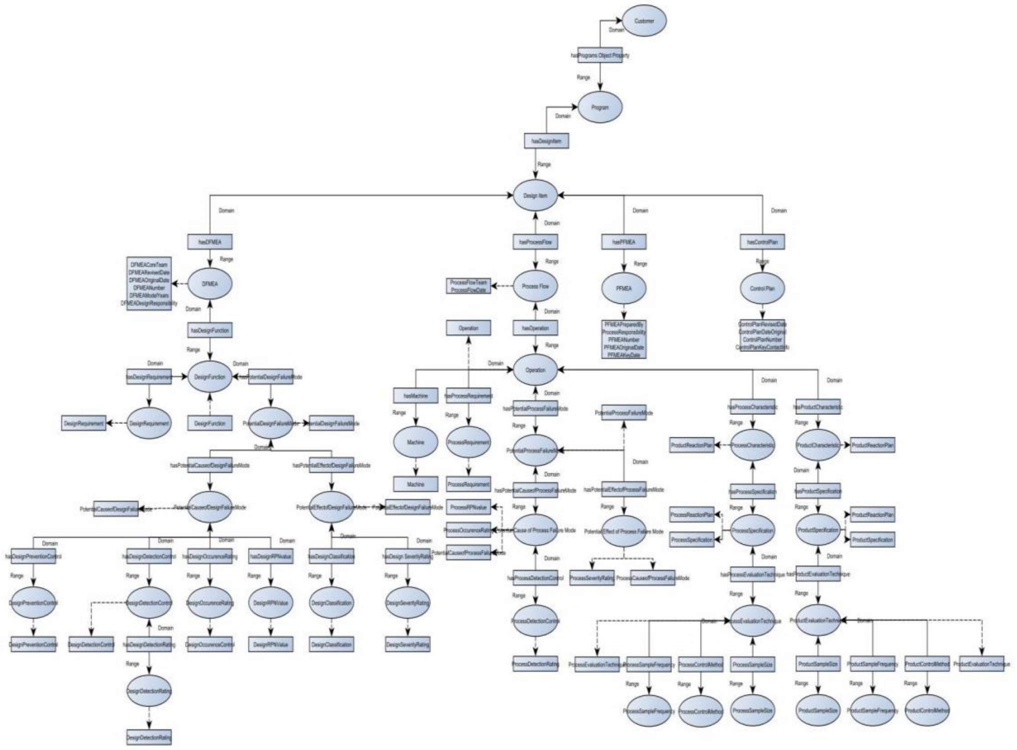


Figure 2-21 Full design and manufacturing ontology from (Chhim et al., 2017)

The CPM (Core Product Model) (see Figure 2-22) in conjunction with the OAM (Open Assembly Model) (see Figure 2-23) both developed by NIST (National Institute of Standards and Technology) form basic "beyond geometry"-level product models that were designed to exist at a level of abstraction that allows them to be capable of capturing common engineering and design information (Rachuri *et al.*, 2006, Fenves *et al.*, 2008). However, these models are in a similar vein to Panetto *et al.*, focussed on the Product Domain with limited consideration of the other domains. The OAM uses a data structure adopted from the STEP (Standard for Exchange of Product data) standard (ISO 10303). STEP is represented by Application Protocols (APs), the most common of which are AP203 and AP214 for the exchange of geometrical information i.e. CAD data, and AP239 for product life-cycle support. Originally STEP was developed within the EXPRESS language, however due to its lack of formal semantics, an OWL-DL implementation of STEP was created and named OntoSTEP (Krima *et al.*, 2009).

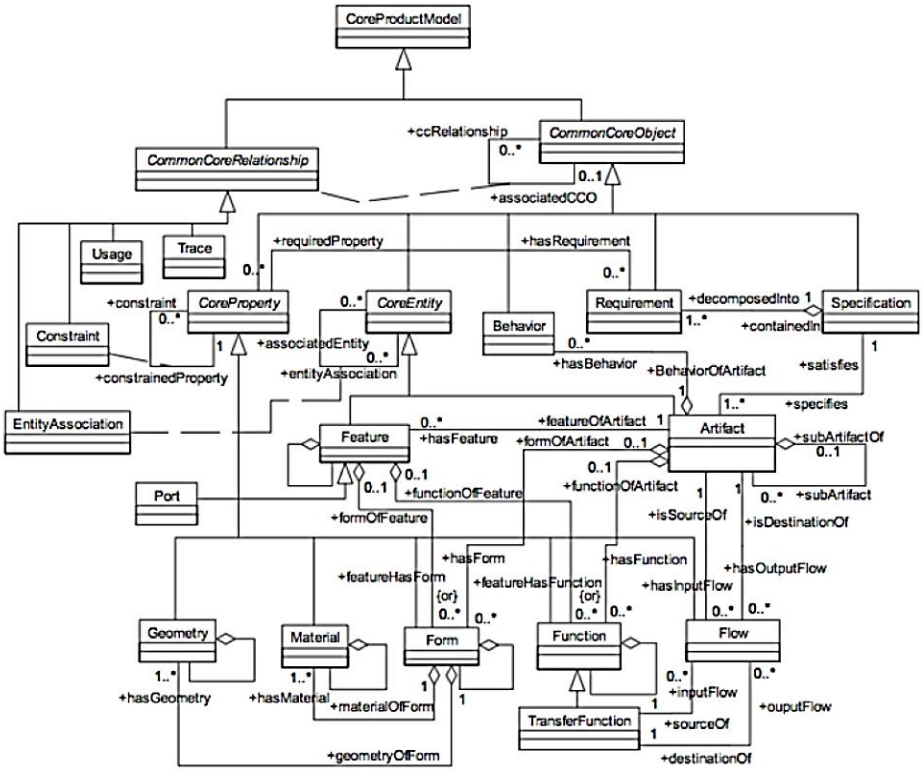


Figure 2-22 The Core Product Model (Rachuri et al., 2006)

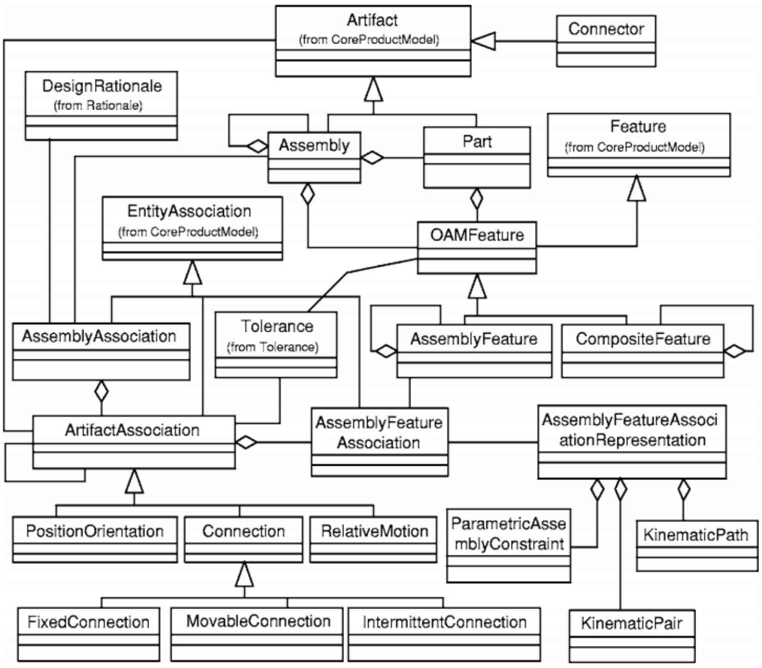


Figure 2-23 The Open Assembly Model ((Rachuri et al., 2006)

Usman *et al.* (Usman *et al.*, 2011, Usman *et al.*, 2013) and Chungoora *et al.* (Chungoora *et al.*, 2013) worked to formalise product and manufacturing concepts in the MCCO

(Manufacturing Core Concepts Ontology) (see Figure 2-24). The aim of the work was to provide an ontological foundation for sharing knowledge across domains. In a similar vein to Panetto *et al.*, the work utilised ISO standards to support in the selection of relevant concepts. A key contribution was the development of a *Feature* concept that facilitated integration. However, there was a lack of modularity in the ontology and the resource concepts did not decompose down to the level of states preventing granular mapping.

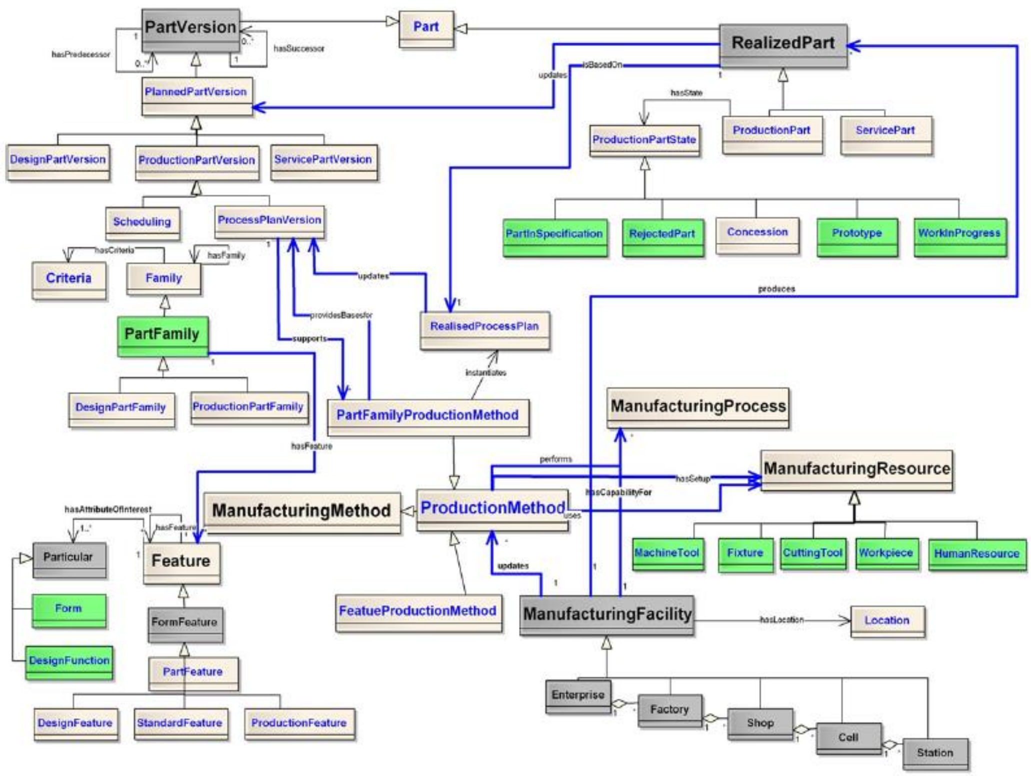


Figure 2-24 Lightweight representation of the Manufacturing Core Concepts Ontology (Usman, 2012)

Within the Process Domain, a commonly cited modelling standard is PSL (Process Specification Language) (see Figure 2-25). The basic concepts represented within the PSL ontology are “Activity”, “Occurrence”, and “Successor”. These concepts together with axiomisation of primitive process concepts provides a rich set of semantically constrained terminologies for describing process knowledge. Although PSL provides a holistic set of capabilities for the Process Domain including process planning, production planning, process simulation, and business process re-engineering (Bock and Gruninger, 2005, Grüninger and Kopena, 2005) it has seen limited development and use. This is largely attributed to a lack of tools and that, in industrial settings, the activities of the Process domain are executed in an ad hoc fashion (Lanz, 2010). The Framework Programme 6 project, PABADIS’PROMISE (Pabadis’Promise, 2006), resulted in the formation of the

P2 model and the P2 ontology. This ontology is holistic as it models all the PPR domains and in a machine understandable way by using RDF. In this work, conversion and transformation of data is executed semi-automatically. Borgo *et al.* (Borgo and Leitão, 2007) define and develop a core ontology for manufacturing. The work utilised an established foundation ontology, DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) to improve system consistency. The work of Borgo *et al.* in conjunction with (Leitao, 2004) created the ADACOR (ADAptive holonic Control aRchitecture for distributed manufacturing control) ontology (see Figure 2-26). This ontology was expressed in an object-oriented frame-based manner and focused on modelling processes and resources with a view to realising changes within the domain of holonic manufacturing systems.

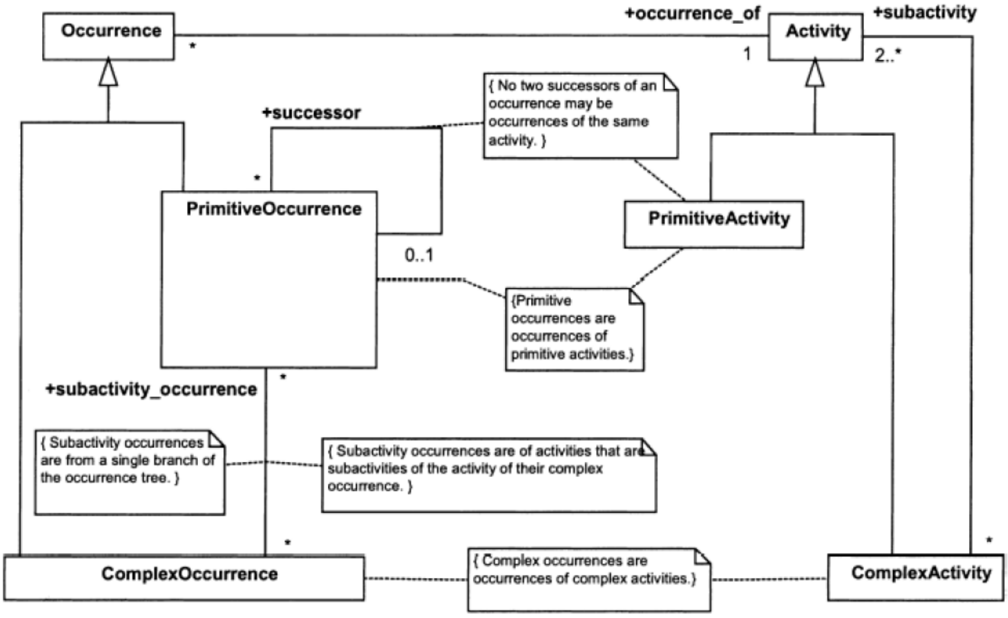

Occurrence
+occurrence_of
Activity
+subactivity
*
1
2..*
{ No two successors of an occurrence may be occurrences of the same activity. }
+successor
*
PrimitiveOccurrence
PrimitiveActivity
0..1
{Primitive occurrences are occurrences of primitive activities.}
+subactivity_occurrence
*
{ Subactivity occurrences are from a single branch of the occurrence tree. }
{ Subactivity occurrences are of activities that are subactivities of the activity of their complex occurrence. }
*
*
ComplexOccurrence
{ Complex occurrences are occurrences of complex activities.}
ComplexActivity

Figure 2-25 Basic Concepts of PSL (Bock and Gruninger, 2005)

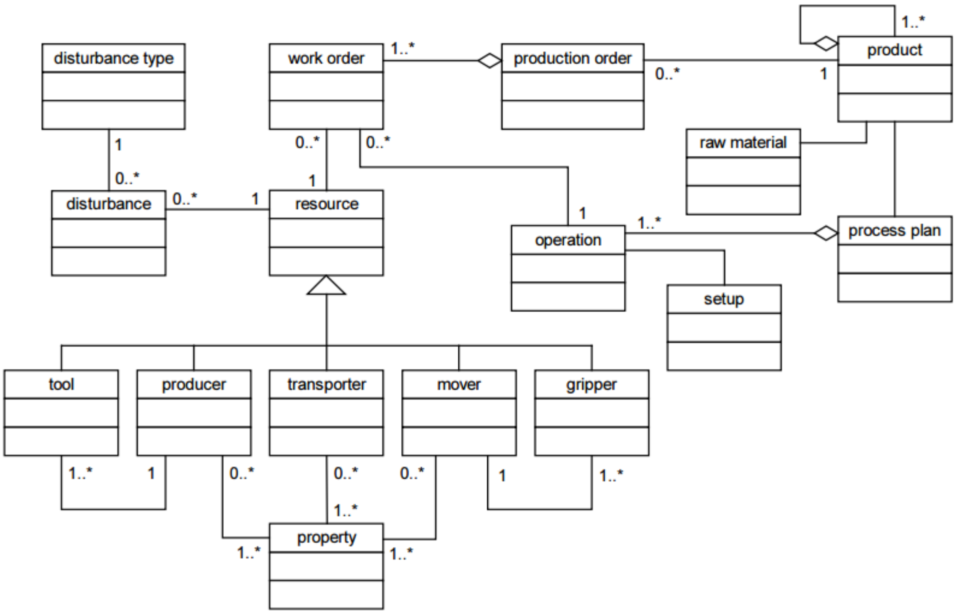


Figure 2-26 Manufacturing Ontology in the ADACOR Architecture (Borgo and Leitão, 2007)

2.5.3 Skill Modelling

The use of a "skill" concept (Pfrommer *et al.*, 2013, Schleipen *et al.*, 2014) and other synonymous terms such as "capability" (Järvenpää, 2012) or "function" (Lohse, 2006) have been used in varying degrees in the literature. One of the earlier examples is in (Oliveira, 2003) which uses a skill concept in an agent-based production system. However, consideration for consistency checking between plans and machine control was not present. In (weser and Zhang, 2009) the appropriate level of abstraction for robot actions is discussed, but inconclusively. The integration of planning and robot control is realised through JShop2, however it was noted that failures in skill execution arose due to inconsistent descriptions. In the ADACOR ontology, the concept of "property" was used synonymously with "skill" however this focused only on the capabilities of the "resource" class and no consideration was made with regards to the needs or requirements of the "product" or "operation" class.

Järvenpää (Järvenpää, 2012) extended the work of Lanz (Lanz, 2010) by enriching the Resource domain model in the PPR modelling framework with a comprehensive capability model to allow manufacturing systems to adapt to new requirements (Figure 2-27). One of the shortcomings of the work was a lack of a clear workflow that demonstrated how the methodology would be used in an industrial context. Furthermore, the work did not consider the program generation or control of machines. Capabilities were abstracted and modelled from the perspective of physical equipment rather than combining them with control aspects e.g. PLC software.

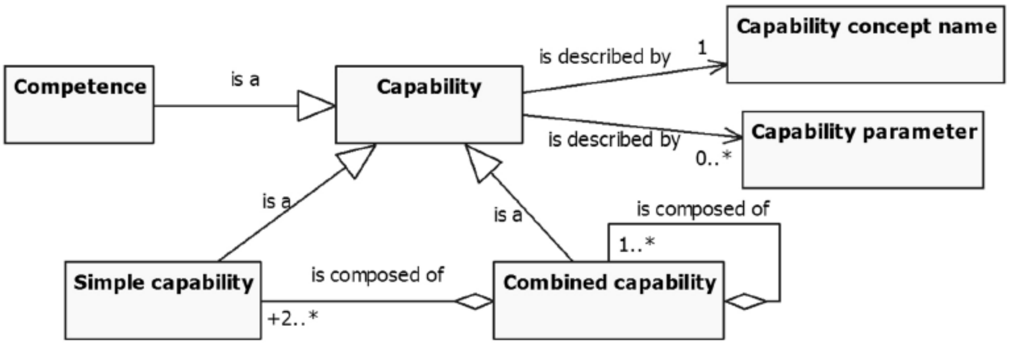


Figure 2-27 Capability model (Järvenpää, 2012)

The lack of use of the ISO 16100 standard was discussed in 2.4.2, however one of the rare instances where the standard is recognised for its potential to support software interoperability is in Matsuda and Wang (Matsuda and Wang, 2010) where it is extended by through a capability template. A matching algorithm was created that matched capability requirements to instances of capabilities that pre-existed within a database. Figure 2-28 illustrates the matching process, where MDM is the manufacturing data model, MSU is a manufacturing software unit, MDD is the manufacturing domain data model, and CCS is the capability class structure. The MDM consisted of a structured activity tree which utilised unambiguous and unique names together with semantic information expressed as a sequence of MDDs. The MDD provided information about resources, processes, information exchange, and resource relationships and would be created by the system designer (those within the Resource Domain. The workflow here is interesting because it is incumbent on the software developer to ensure that the software unit they develop aligns with the model so that the end user can query for it accordingly. This enforces alignment between the teams and stakeholders that are involved with realising a complex system. As the system was implemented within a .Net Framework using C# mapping the aforementioned benefits of ontologies could not be exploited. Furthermore, the work did not actually show any integration with engineering software despite the claim in the title i.e. "Software **interoperability tools**..." This lack of integration prohibited the work from being validated within a workflow that could be aligned to industrial practices.

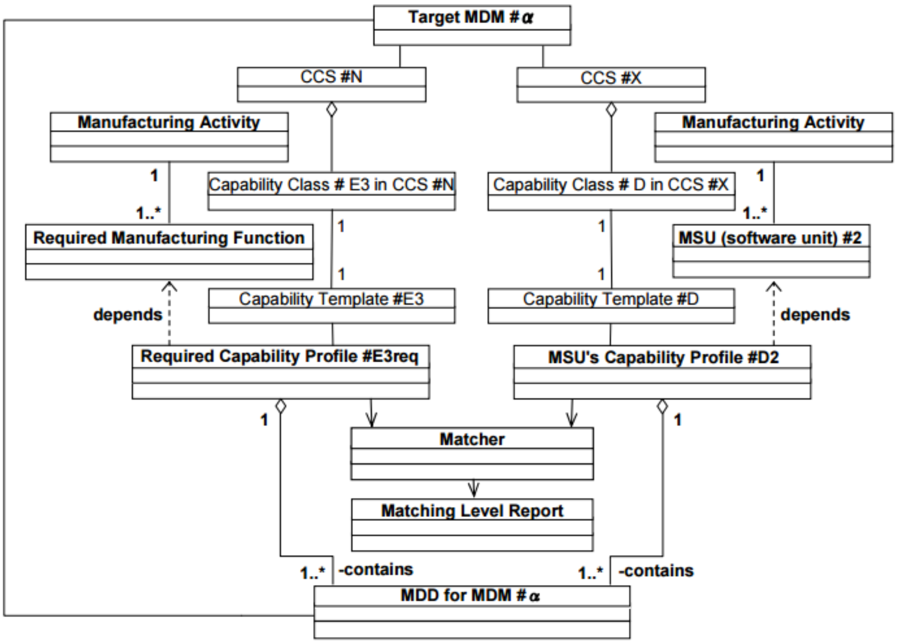


Figure 2-28 Matching software capability profiles (Matsuda and Wang, 2010)

As alluded to in the section that discussed the digitisation of manufacturing, the shift towards smart manufacturing facilitated by technologies such as the IoT and driven by ever-challenging customer requirements has resulted in the emergence of Cyber-Physical Production Systems (CPPS). Within the context of system operation, skills are executed by “agents” or encapsulated within “services” (Leitão, 2009, Stark *et al.*, 2017). There remains a disconnect however between the functional capabilities of a CPPS, the representation of those capabilities within digital models, and the execution of capabilities. The IEC 61499 (Vyatkin, 2009) function block standard is a modelling approach that has the potential to bring these elements together, however there remains a lack of uptake by major PLC vendors, primarily due to the domination of IEC 61131 (Hanssen, Leitão, 2009). An example of the capability of IEC 61499 is described in Alsafi and Vyatkin (Alsafi and Vyatkin, 2010) where it is used to support software reconfiguration through an ontology-based reconfiguration agent. Although there is a lack of visualisation to validate changes to the system control, the work demonstrates how the knowledge model infers facts about the manufacturing environment and then decides whether a set of requirements can be met. One of the key focus points of this thesis is the mapping of high level process planning with low level mechatronic control and this is also addressed in (Alsafi and Vyatkin, 2010). The Evaluation chapter discusses in more detail how the work in this thesis complements and extends the relevant aspects of Alsafi and Vyatkin (Alsafi and Vyatkin, 2010).

There have also been a number of EU funded projects in the last decade that consider skills within their scope, these are summarised as follows:

- SIARAS (Skill-based Inspection and Assembly for Reconfigurable Automation Systems) built a skill-based model to connect top-down and bottom-up views on the system reconfiguration process (Malec *et al.*, 2007, 2008, Haage *et al.*, 2011) (Figure 2-29). An ontology of skill primitives was developed within a skill server to aid the matching of process requirements to resource capabilities.

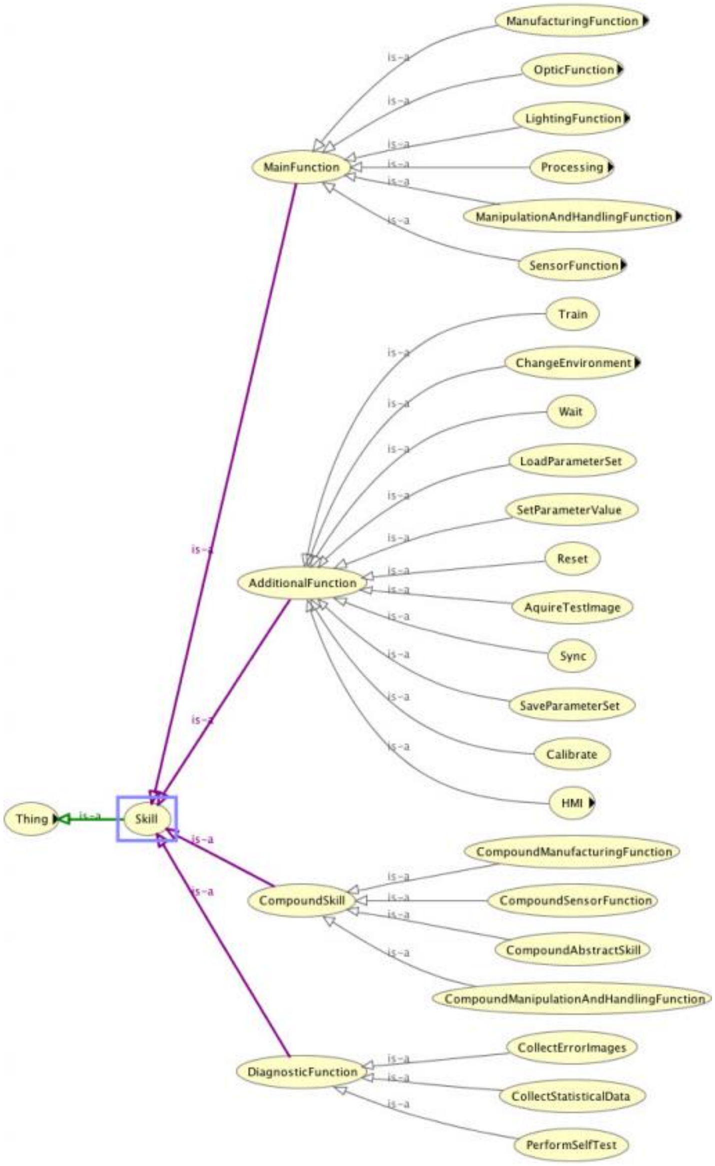


Figure 2-29 Top skill classification, as defined by the SIARAS ontology (Stenmark and Malec, 2015)

- ROSETTA (Robot control for Skilled ExecuTion of Tasks in natural interaction with humans; based on Autonomy, cumulative knowledge and learning) used skills to execute tasks in environments where robots interacted with humans (Björkelund *et al.*, 2011b, Björkelund *et al.*, 2011a). The approach used AutomationML as a data exchange format and build a knowledge base by converting this structured data into RDF triples. The work resulted in the culmination of the "Knowledge Integration Framework" (KIF) which exploited the skill and device ontology that had been developed (see Figure 2-30). The KIF server was at the centre of an architecture consisting of a production station, controller, and various interfaces with implementation carried out within Robot Studio software by ABB. One of the limitations of the work was the syntax-based translation of XML files which does not align with the broader vision of the semantic technologies used. Furthermore, due to the focus on capturing execution/operational data and converting it to skill knowledge, there was a lack of description as to how this knowledge would be mapped with other domains.

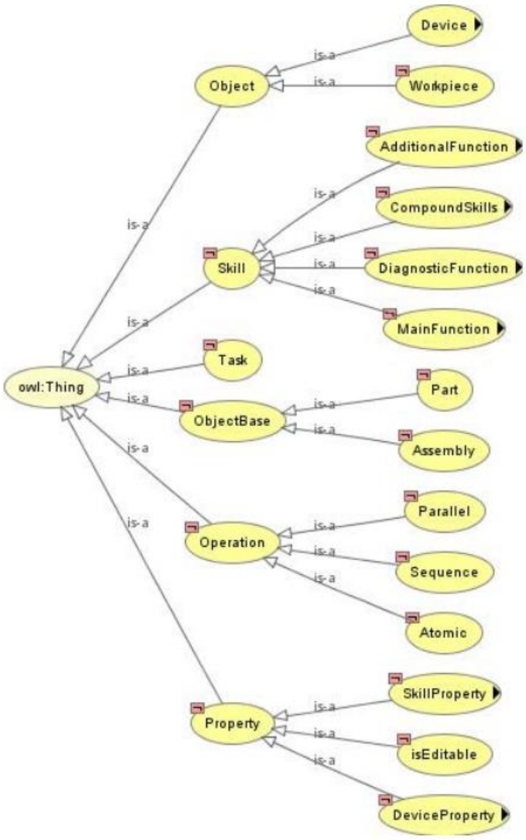


Figure 2-30 Top level ontology used in the ROSETTA project with aspects of a Skill model (Björkelund et al., 2011b)

- IDEAS (Instantly Deployable Evolvable Assembly Systems) focused on the implementation of agent technology (Onori *et al.*, 2012). An evolvable assembly system (EAS) is one that co-evolves with the evolution products and processes (Oliveira, 2003). The project took advantage of a number of developments from the EUPASS FP6 project, namely: ontological descriptions of assembly processes, equipment modules with embedded control, and data exchange protocols. The implementation of agent technology was achieved by exploiting the IEC 61499 (Vyatkin, 2009) standard for distributed control to enable the vision of "plug & produce" (Ferreira and Lohse, 2012, Ferreira *et al.*, 2012). This required the development of a skill model which consisted of "atomic skills" that were defined at the lowest level of granularity or at the module level, and "composite skills" which aggregated module "atomic skills" to form more complex skills. The definition of a skill consisted of four main characteristics: assembly process type, level of granularity, control ports, and parameter ports. Figure 2-31a is a schematic overview of the composite skill concept using IEC 61499 notation. There is a high level of similarity between this work and Wang *et al.* (Wang *et al.*, 2008, Wang *et al.*, 2012) with both focus of the execution of skills using function blocks.

  In order to realise the "evolution" aspect of the approach a configuration process was developed. This was also supported through the skill approach with the workflow illustrated in Figure 2-31b. First the assembly process requirements need to be defined based on the new product/product variant assembly step sequence, precedence constraints, and process parameters. Next, skills are assigned to the assembly process requirements. Finally, the third step considers those requirements that are not executable by the existing skills necessitating the generation of new ones.

  The work showed potential in the sense that the capabilities of a system could be described within a rich, extensible model. Furthermore the project successfully demonstrated multi-agent control for assembly systems by building a number of physical demonstrators. However, the approach did not use semantic technologies preventing, for example, the querying of skills at the product development stage to ascertain what change, if any, would need to be made to the system. As such, the workflow described would be largely manual or possibly implemented within a database resulting in poor extensibility. Furthermore, the skill model did not explicitly link with Product and Process Domain activities, sitting squarely within the Resource domain as properties of modules.

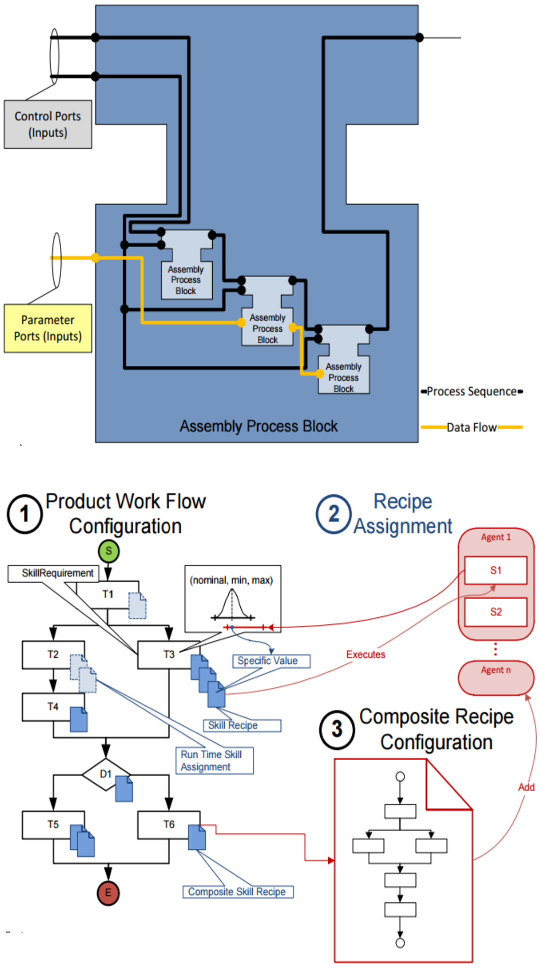


Figure 2-31a) Overview of the composite skill concept, and b) conceptual overview of the configuration process (Ferreira and Lohse, 2012)

- SkillPro (Skill-based Propagation of "Plug&Produce"-Devices in Reconfigurable Production Systems by AML) (Pfrommer *et al.*, 2013, Schleipen *et al.*, 2014) utilised the PPR concept that exists within AutomationML (Drath *et al.*, 2008) with the addition of production component skills to develop a holistic service-oriented framework for adaptable production systems. In the SkillPro project, the "Skill" is a placeholder for a process and provides metadata such as parameters needed to specify it. A skill hierarchy exists thus building a skill taxonomy however it appears that the creation of such a taxonomy was beyond the scope of the project as this cannot be found in the literature. A "Production Skill" provides some indication of production requirements and seems to be an output from process planning activities, while the "Asset Skill" is the skill that is executable by a physical asset. In order to manage assets, an asset management system was developed supporting the paradigm of digital manufacturing as a library that can be reused. It was not clear how this asset management system was implemented and thus how it would integrate with engineering software. One interesting extension of the skill model in SkillPro that has not been observed in other works is the recognition of the skills that human operators have. The skill concept and its relation with the PPR model is illustrated in Figure 2-32. The work did not use did not use ontologies it suffers from an inability to reason about skills and inconsistencies across domains or infer new knowledge from explicit relationships. Furthermore, there was limited evidence of the models themselves resulting in the output of the project seeming more conceptual in nature and thus difficult to evaluate.

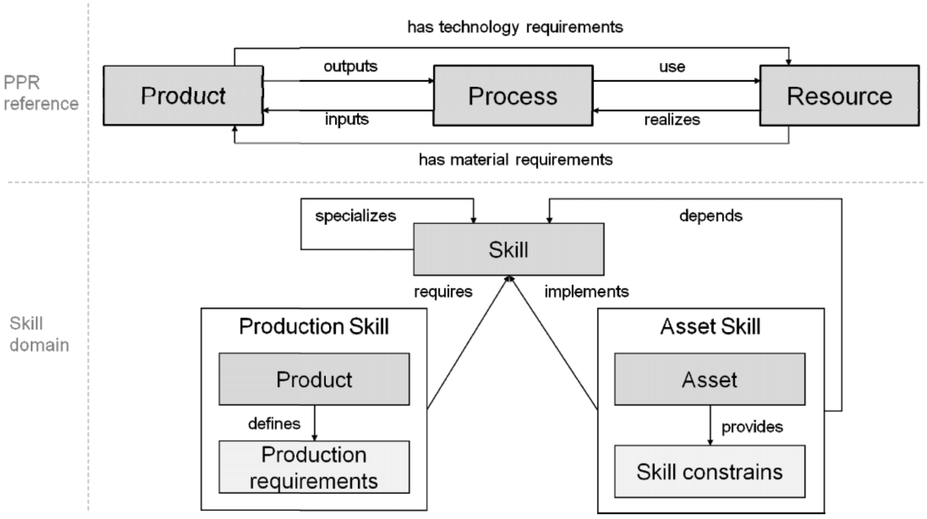


Figure 2-32 Skill concept aligned to classical PPR from the SkillPro Project (Aleksandrov et al., 2014)

- PERFoRM (Production harmonized Reconfiguration of Flexible Robots and Machinery) is an ongoing project (at the time of writing) that continues work on the concept of “plug & produce”. Ontologies are to be used within the Resource domain to facilitate interoperability of heterogeneous devices (Leitão *et al.*, 2016).

2.6 *Summary and Gap Analysis*

In order to support the paradigm shift towards mass customisation and reduced product lifecycles, engineering changes must occur, and must occur more frequently, through a number of engineering domains fluidly. Thus, this chapter opened with describing change propagation and the engineering change management process presenting the challenges highlighted in the literature. The review found that increasingly complex products in conjunction with a more demanding customer base (a consequence of the paradigms of mass customisation and personalised production (Mourtzis and Doukas, 2014)) increases both the complexity and the frequency of the engineering change process. Change propagates to a larger number of stakeholders and through more means than was conceivable a few decades ago.

Although in principle model-based systems engineering provides a business the capability to have defined relationships across multiple design and engineering domains and phases, models are often created in proprietary standards which are industry or software vendor specific. As a consequence, despite scientific and industrial research efforts, models remain disjointed and uncoupled in many instances. Furthermore, a typical outcome of MBSE is the emergence of conflict as inconsistences arise between different models that, regardless of the level of activity decoupling, need to be addressed. This is referred to in the literature as inconsistency management and the literature review explores the different methods associated with identifying and resolving this issue. Examples of this have been presented with ontologies, but without reference to any skill models.

Within the review, it was identified that there are some connection points between the Product Domain and Resource Domain i.e. through the exchange of information concerning geometries. Logical aspects such as sequencing and selecting the appropriate resources based on process types is achieved through activities classically associated with APP. The argument presented in this section is that the existing methods and tools are not connected in a way to the other domains that allows changes to be communicated in an effective way. Industrial approaches are manual and therefore error prone, and even digital models while providing a degree of stability and traceability are often not integrated with other domains. There is a need for humans to interrogate documents or models is changes are made to the Product domain to understand how the process plan will change and inform the Resource domain accordingly. ***There are some examples of automating the change presented, but a holistic methodology is missing.***

To address the integration and interoperability, the remainder of the literature focused on how ontologies have been used to formalise semantics and store the knowledge that currently exists in the minds of experts. Within the context of ontological models, there many examples and some level of convergence can be seen with respect to the concepts, but there remains a disconnect as to the semantics or definitions between the models of different authors. There was a detailed section on skill modelling as a means for representing the functional capability of manufacturing system resources and how such models are used to connect the respective product realisation domains. Ultimately however,

the focus of skill models remains on the execution aspect of manufacturing system with limited consideration for how the workflow to realise change is supported.

2.6.1 Knowledge Gaps

Based on the review of literature, the author identifies the following knowledge gaps:

- Both ontological models of the respective PPR domains and those for skill models exist in the literature, however where the two exist in a single piece of work, the latter always sits squarely in the Resource Domain with limited explicit consideration for its interaction with the other domains. ***There is a lack of knowledge as to how the Product Domain and the Process Domain interact with skill models and how this connection can support in inconsistency management within the context of engineering changes.***
- One the significant benefits of ontologies is their ability to enable interoperability between heterogeneous systems. Within the context of manufacturing systems research, this is largely focused on enabling the interoperability of heterogeneous devices and in some cases executable software e.g. the PERFoRM project. However, there is a lack of knowledge as to how ontologies (in conjunction with skill models) can support in the engineering workflow associated with engineering software, particularly those used for the design and visualisation of manufacturing systems. ***The interaction between ontologies and software tools has not been explored in considerable detail beyond the former's ability to store the knowledge generated by the latter, and not how said knowledge can be exploited and reused in a practical way.***

The following chapter address these knowledge gaps by first identifying the key concepts that should exist within the PPR domains based on what has come before, the formation of a Skill model that brings the domains together, and a broader framework that facilitates the integration of engineering workflows with knowledge representation.

3 A knowledge-based approach for integrating engineering workflows

3.1 *Introduction*

The gap analysis at the close of Chapter 2 identified that PPR approaches using ontologies have been demonstrated in the literature, but there is limited evidence of how these models can be used to complement existing engineering workflows. In addition, the use of deduction and inferences through the workflow is limited and there are only a few examples of a how Skill or Capability models facilitate design, development, and modifications across the PPR domains. Therefore, the vision for how the PPR models (in conjunction with the Skill model) fit into the wider workflow at a high level is presented in Figure 3-1. Domain stakeholders work within their respective teams using their respective engineering tools/methods and modelling software with a “team layer”. However, more often than not, the connectivity and integration of the digital models is poor (and integration of paper-based documentation is purely manual). Thus the communication of requirements and constraints are carried out in an informal manners e.g. through meetings, emails, documents. This research proposes an addition to the workflow of the team layer through the Skill model which extends the descriptions of the digital models such that they can be effectively integrated into the knowledge layer i.e. the PPR ontology. The digital models could be parsed through a standard like AutomationML, however to ensure consistent semantics and the explicit declaration of contexts, the Skill model is fundamental.

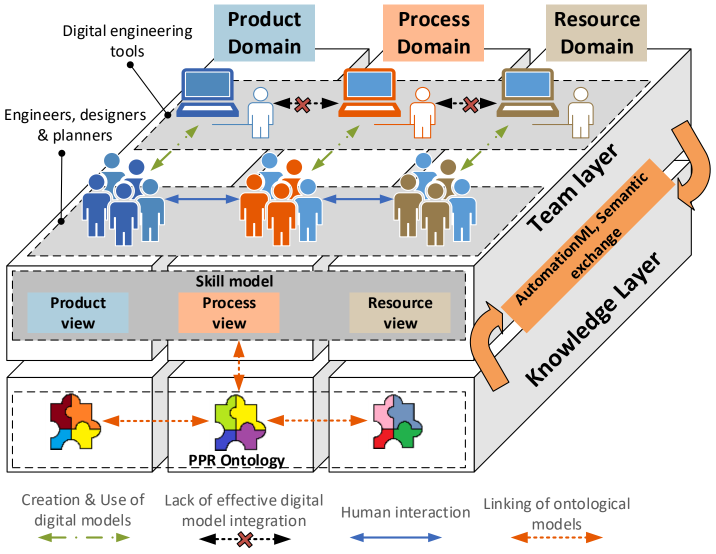

Digital engineering tools
Product Domain
Process Domain
Resource Domain
Engineers, designers & planners
Team layer
AutomationML, Semantic exchange
Knowledge Layer
Skill model
Product view
Process view
Resource view
PPR Ontology
Creation & Use of digital models
Lack of effective digital model integration
Human interaction
Linking of ontological models

Figure 3-1 Lack interoperability and knowledge integration addressed through PPR ontology and Skill model

3.2 *Methodology overview*

This chapter begins with a general description of the model used for this approach. It then presents the respective PPR domain ontologies justifying the concepts, the descriptions, comparing the structure and semantics with existing works and finally presenting the intra and inter domain relations. Once the ontologies have been described, their use to support inconsistency management as a consequence of engineering changes is discussed. In Daconta *et al.* (Daconta *et al.*, 2004) three representation levels for ontologies are described as a structured method for implementing knowledge representation. These are described as follows:

- At Level 1 there is the Knowledge Representation (KR) level which includes the fundamental constructs associated with KR such as Classes, Axioms, Rules etc. In this work, the Web Ontology Language (OWL) is used which is an enriched extension of the Resource Description Framework (RDF) language. The environment used to create and edit the ontological models is Protégé which is an ontology editor developed by Stanford University .
- At Level 2 there is the Conceptualisation process which identifies the necessary concepts needed to capture the part of reality being modelled in the ontology (Guarino, 1998). The implementation of KR within modular ontologies in this work is described using standard domain terminology. Standard terms have been derived from the literature, with key sources being ontological models with a similar focus, namely: (Lanz, 2010, Lastra, 2004, Järvenpää, 2012, Delamer and Lastra, 2006, Lohse, 2006, Usman *et al.*, 2013, Panetto *et al.*, 2012) as well as some standards that exist, particularly within the context of assembly such as DIN 8580 (DIN, 2003) and VDI 2860 (VDI, 1990). These references have been examined in Chapter 2 and the reader is directed there for further reading as well as the sources themselves. When describing the respective domain ontologies and the chosen concepts, the author justifies why a given concept has been chosen over another. In some cases the choice is “just” semantics. In other cases, there is a significant impact on the topology of a given domain and its relationship with others. The conceptualisation level is represented through UML class diagrams (though not using strictly formal syntax) as it permits the elaboration of data type properties, multiplicity, and relationships such as aggregations and sub-classes.
- At Level 3 there is the Instantiation of the ontology which is the process of populating the classes with individuals. In this research work this process is carried

out as a means to test the strength and capability of the conceptualisation level. With respect to the research questions, knowledge gaps, and the contributions, the tests determine:

i) whether the domain models sufficiently covers the range of concepts needed,

ii) how capabilities are checked for using the Skill model

iii) how inconsistencies are identified

iv) how control code for machines can be modified based on the identified inconsistencies.

There are a number of challenges in determining whether an object should be conceptualised or instantiated. It is essentially dependent on the level at which the ontology is to be used and how it is to be used. The distinction between Level 2 and Level 3 has been made based on this consideration.

The PPR ontology consists of a model of three main modules i.e. Product, Process and Resource as well as a Skill model, which include cross domain links and rules to infer implicit knowledge. The result is an ontology with sufficient level of description that can be used for supporting the re-engineering process when introducing new product variants in assembly lines. As the focus is the link between the domains and the assembly process planning activity, there is a heavy focus on this area. That is not to say that the ontology will not have other uses also. These are investigated and considered in Chapter 5 – Evaluation.

Moreover, the structure of the resource ontology is aligned with use of virtual engineering (VE) tools called vueOne. The engineering toolset used are described in detail in (Harrison *et al.*, 2016) but can be summarised as a lightweight, low-cost toolset that aims to complement commercially available VE solutions. vueOne used standards and open data formats that allow interfacing with other engineering environments. However, the modular nature of the ontology permits the addition of concepts that may not yet exist depending on the desired tool to be used. During the development of the ontology, a number of shortcomings of the engineering toolset were identified. In brief this included a lack of detailed product modelling, limited high level process planning, and complex change processes should models need to be rectified. The methodology chapter describes the “as is” state of the engineering tools and what extensions are proposed to support implementation of the research work with a view to providing a generalized set of recommendations for industrial software also.

3.3 *Domain Ontologies*

At their highest level, the domain ontologies in conjunction with the skill model can be represented as presented in Figure 3-2. The Product domain, in order for it to be realised requires some set of processes that are described by the Process domain. There are a number of works in the literature that have been presented in Chapter 2 that automate the process of converting a set of product component relations into an ordered set of processes.

There is some inherent knowledge that could be stored and referenced as a consequence. The objective of the link between the Product Domain and the Process Domain is not the storage of general knowledge i.e. that an instance of a screw in the Product Domain would require an insertion process in the Process Domain, but specific knowledge that would be generated through industry or organisation specific workflows e.g. that a specific product assembly is realised by a specific operation.

The knowledge of what that operation is, could either be stored in the ontology or instead it would refer to a location where such information could be found. This prevents the ontology becoming heavy and thus computationally intensive to use as well as minimising data duplication. On the other hand, there are provisions in the ontology to store knowledge of what a given operation consists of should that be useful to the user. This offers flexibility in its usage and thus does not impose a specific way of working on an organisation, complementing workflows or engineering processes that may already exist. Both the Product Domain and the Process Domain point towards the Skill model.

The Product domain will, in general, provide the model with some context i.e. what is it handling? how heavy is it? what is the material? On the other hand, the Process domain provides the requirement of the action i.e. I need to grip. I need to rotate. I need to move etc. The issue of semantics arises here and this is conceptually addressed by proposing that a standard terminology should be used. This is expanded upon in the Process Domain ontology section.

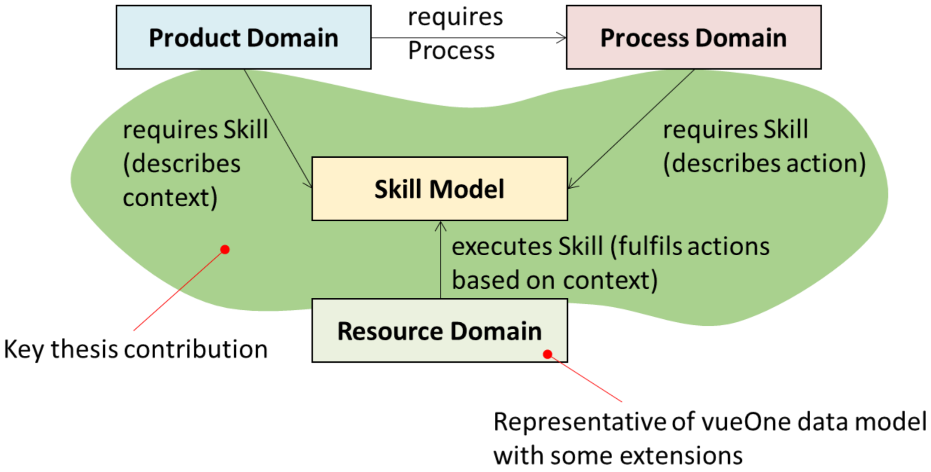

Product Domain
requires Process
Process Domain
requires Skill (describes context)
Skill Model
requires Skill (describes action)
executes Skill (fulfils actions based on context)
Resource Domain
Key thesis contribution
Representative of vueOne data model with some extensions

Figure 3-2 Model Overview with contribution

Finally, the Resource Domain will, in general, execute a Skill. The key difference between the ontological decomposition of how the Resource Domain executes a Skill as compared to other similar works e.g. (Aleksandrov *et al.*, 2014, Björkelund *et al.*, 2011a, Björkelund *et al.*, 2011b, Pfrommer *et al.*, 2013, Pfrommer *et al.*, 2014, Schleipen *et al.*, 2014, Järvenpää *et al.*, 2010, Järvenpää *et al.*, 2016) is that it is based at a much finer level of granularity. Conventionally, other authors have stipulated the execution of skills or capabilities at the machine or station level. Although this is acceptable if a high level understanding of what a station or machine is able to do is required, this does not a allow the more nuanced behaviour of such equipment to be represented. This returns to the industrial issues associated with engineering change management. If changes are made at a high level without the associated knowledge of the impact of what happens at a finer level of detail, such change processes are likely to face unforeseeable hurdles due to the lack of models and thus transparency.

The domain ontologies are modularised to demonstrate how information can be linked and exploited should, as is sensible to expect, respective domain knowledge structures be designed by domain experts. Similar insights and approaches have been made and used by Lohse (Lohse, 2006) and Ramis Ferrer *et al.* (Ramis Ferrer *et al.*, 2016). Ensan and Du (Ensan and Du, 2011) discuss the challenges of monolithic ontologies to be not only maintenance, reasoning, and implementation due to their complexity, but also the inability to work in a distributed environment, which is commonplace for modern manufacturing organisations. The encapsulation of knowledge into an ontology module defines the content as well as the interfaces or ports to other ontologies permitting a given system model to be used from perspectives that may not have been considered at the time of design. Note that this can also be achieved through defining interfaces or ports within an upper ontology. This plays well into the idea of ontologies being extensible as opposed to the more rigid nature of relational databases (Martinez-Cruz *et al.*, 2012). Within the context of PPR modelling, the boundaries that exist in the literature concerning which activity or concept should be in which domain is hard and clear, however this cannot be expected to be the case in every industrial setting. Therefore, certain concepts need to be shifted and plugged into other areas to be aligned with specific industrial domain needs. Encapsulation of certain aspects of knowledge within domain ontologies facilitates the shifting of broader concepts where there is certainty and thus allows a KB that is more representative of a given organisation's operating structure, to be created.

Having provided an overview of the models and their relations, the chapter progresses to a more detailed description of the respective domains. Where possible, the author has attempted to ensure consistency of colours associated with domains to make diagrams easier to follow. The Product Domain is blue, the Process Domain is red, the Resource Domain is green, and the Skill model is yellow.

3.4 *Product Domain*

The focus of the methodology as a whole is on assembly processes and therefore concepts within the Product Domain align with this focus. Many of the concepts and relations within this domain are based on the works of Lohse, Kim (Lohse, 2006, Lanz, 2010, Kim *et al.*, 2006, Demoly *et al.*, 2010, Fenves *et al.*, 2008, Technology, 2005) where they have also considered the structure of the product concept. In this work, the product model is focused on ensuring that information about the broadest breadth of an organisation's product family can be captured, the features associated with product components, and the relations between product components and assemblies. The product design process and associated information is not included in this work, only the results of this activity. Furthermore, there is no representation of the product or component geometry within the ontology. This information is abstracted away as the ontology would be better served as a mechanism for pointing to the file/model associated with detailed topology.

3.4.1 Modelling Product Variety

The *ProductFamily* is a high level concept that allows the representation of a broad range of products that may exist within an organisation. A *ProductFamily* contains a set of *ProductVariants* that share a common set of attributes. The *ProductVariant* is realised through a set of *Operation* instances through the *hasOperation* property which is a link to the Process Domain ontology. This is elaborated upon in the next section. A *Product* is defined by the business dictionary as “A good, idea, method, information, object or service created as a result of a process and serves a need or satisfies a want. It has a combination of tangible and intangible attributes that a seller offers a buyer for purchase .” This is a broader definition of *Product* than is necessary for this work as products such as services can exist entirely within a digital environment and so negate the need for physical assembly.

However, the author chooses to retain this definition due to it explicitly defining a link to process and the mention of it serving a need or want. While this research does not extend the Product Domain ontology to an area of what the product does, it would be a useful addition to connect the ontology to the market i.e. external to the business environment where such a model would be used. In order to help the reader understand how the concepts of *ProductFamily, ProductVariant,* and *Product* would be instantiated, example from a number of different manufacturers are presented in Table 3-1.

Table 3-1 Example of Product, ProductFamily and ProductVariant

Industry	**Concept**		
	Product	*ProductVariant*	*ProductFamily*
Automotive	Jaguar XF R-Sport, LHD, 250PS, Auto, Black, Gasoline	Jaguar XF	Jaguar
Electronics	Samsung Galaxy S8, Midnight Blue, 64GB	Galaxy Series	Mobile Phones
Fuel Cell	Open Cathode AC64, 2kW	Open Cathode fuel cell	Fuel Cell

Modern products are more complex now than they have ever been and the level of customisation means that the hierarchy presented in this ontology may prove to be insufficient to capture the depth that may be required. This could be alleviated in part by introducing new concepts above *ProductFamily* to support in the level of steps that may be required by an organisation. Equally, for the sake of simplicity, it may not be necessary to utilise the three levels presented and the user may only instantiate to the level of *ProductVariant.*

3.4.2 Assembly

The *Product* is composed of any number of assemblies. In this context, the word *Product* is synonymous with "final assembly". The term *Product* was chosen over final assembly because there are no examples of the latter being used in existing ontologies. Furthermore, it is the *Product* that the customer receives in every instance and not a final assembly. The difference may be some end of line testing or packaging. The *Assembly* is an aggregation of *ProductComponents* in a way that respects the *Liaisons* between them. A *Liaison* is defined as "the physical connection that exits between two components within an assembly" (Lohse, 2006), and within the context of this ontology this refers to the *ProductComponent* and the *Assembly*. Note that the concept of *Liaison* can also exist within the Resource Domain. However, in that domain, this is avoided due to a functional view being taken of the system due to a "component-based" philosophy being employed by the author in that domain. In other words, the physical relationship that exists between components in the Resource Domain is less relevant than the functions of *Skills* that they are able to execute. Should knowledge be required concerning physical connections, this could be derived from the virtual model.

The concept of sub-assemblies is handled through the *contains* object property. Both the *ProductComponent* and the *Assembly* are connected to the *Liaison* class through *hasLiaison* object property. The approach for modelling the *Assembly* in this way is based on the work of (Lohse, 2006). This is due to it being a proven solution for describing product components with respect to the assemblies they form and the relationship between both components and assemblies. An *Assembly* is considered to be an undirected graph in the Product Domain with *ProductComponent* and *Liaison* representing nodes and edges respectively.

An illustrative example using a fuel cell is presented in Figure 3-3 that shows an exploded view of the product on the left (a) labelled with component names. Figure 3-3b illustrates how this information is transformed into a graph. Note that each of the liaisons has been given a unique name which aligns with the general approach of OWL and Semantic Web Technologies revolving around Unique Resource Identifiers (URIs). This explicit declaration and thus the ability to directly instantiate relationships between components is more expressive than modelling relationships through object properties as is the case in (Ahmad *et al.*, 2015b). However, naming liaisons is not typical practice in industry as it requires the management of an additional data set. Therefore it could be possible to auto-generate liaison instances through the aggregation of the names of the product components or assemblies involved with a given liaison.

Only three types of *Liaison* are modelled in this work, but this can be extended by adding further classes due to the extendable nature of ontologies. In this model, the *Liaison* has been given a data property value of *hasLiaisonQuantity* to determine how much of a given *Liaison* exists which helps to identify whether the resources are capable of meeting requirements. The uses of this data property value could be: generation of cycle times if a time value is assigned to a given *Liaison*, to provide a mechanism to check product designs with previous variants, to ensure that the Process domain has ensured that all liaisons are realised in the process plan. The realisation of a *Liaison* requires a *Process*, while the *Operation* may be specific to a *ProductVariant*. The *ProductComponent* class has data property values of *hasProductComponentQuantity* and *hasProductComponentID.* These data properties can support in the management of product bill of materials. An *Assembly* and a *ProductComponent* are also both examples of a *SkillContext*. This concept is discussed in further detail in when presenting the Skill model.

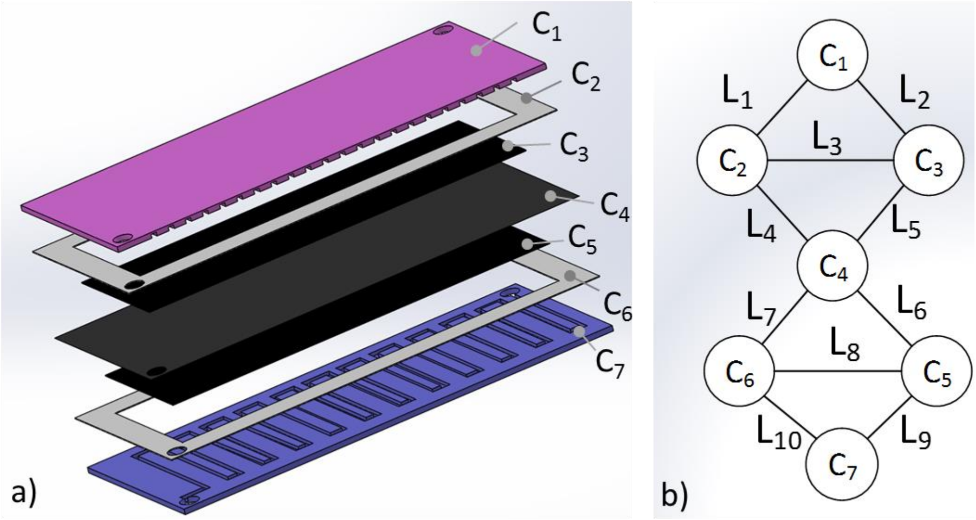


Figure 3-3 a) exploded view of fuel cell b) undirected graph of fuel cell assembly (Ahmad et al., 2016)

3.4.3 Features

Although there is no model representing detailed product geometry, there are certain features of a *ProductComponent* that can be represented through the *ComponentFeature* class. The use of feature models within the Product domain has also been included in the works of Usman *et al.*, Lanz, Kim, and Demoly (Usman *et al.*, 2013, Lanz, 2010, Kim *et al.*, 2006, Demoly *et al.*, 2010, Technology, 2005). The author has taken inspiration from the model presented in Lanz (Lanz, 2010) which decomposes features into Geometric and NonGeometric. In this work, the *ComponentFeature* has two subclasses that are *QualitativeFeature* and *QuantitativeFeature.* The former concerns those features that cannot be described through integers. These include colours and materials. These concepts have been defined as classes rather than as instances of the superclass *QualitativeFeature* as there is knowledge that needs to be represented at this level to support the selection of appropriate manufacturing resources. For example, a component within a fuel cell is the gas diffusion layer (GDL). This component is made from carbon paper which is a porous material. Therefore, the author envisions a material ontology e.g. Ashino and Fujita (Ashino and Fujita, 2006) that could extend the Product domain and enhance the Skill statement i.e. increasing the breadth of information available to the Skill model to ensure that the appropriate resources can be selected. Upon selection of appropriate equipment/resources, the knowledge associated with the selection process could be stored explicitly as a triple. This could then be used to infer appropriate resources when the same material is used in a different context.

The *QuantitativeFeature* is modelled in a different way to *QualitativeFeature* to exploit the fact that this type of feature can be expressed through integers. The data type property of OWL is used to model the *QuantitativeFeature* and this class can quite easily be extended by adding new properties. The author has elected not to represent these properties as classes because there is little else that can be gleaned or inferred from this information. As an instance of *ProductComponent* is a physical thing it goes without saying that it will possess some physical attributes.

There is no use case that the author has been able to identify (within the context of assembly automation) that would lend itself to infer, for example, that a robot will need to lift an object of mass. Rather, it is the value of the mass associated with the object that is important and this cannot be captured through the use of a class. If the ontology was to be extended and fully align with the definition of product presented at the head of this section, then there may well be a need to transform the data properties associated with representing *QuantitativeFeature* to classes. This is because non-tangible products will not have physical attributes e.g. a mobile phone app. This would require an extension of the *ProductComponent* class to represent tangible/physical and non-tangible components. In this case it could be useful to infer that a component does not have a mass associated with it and therefore there is no need to make inferences concerning it from physical parts of the

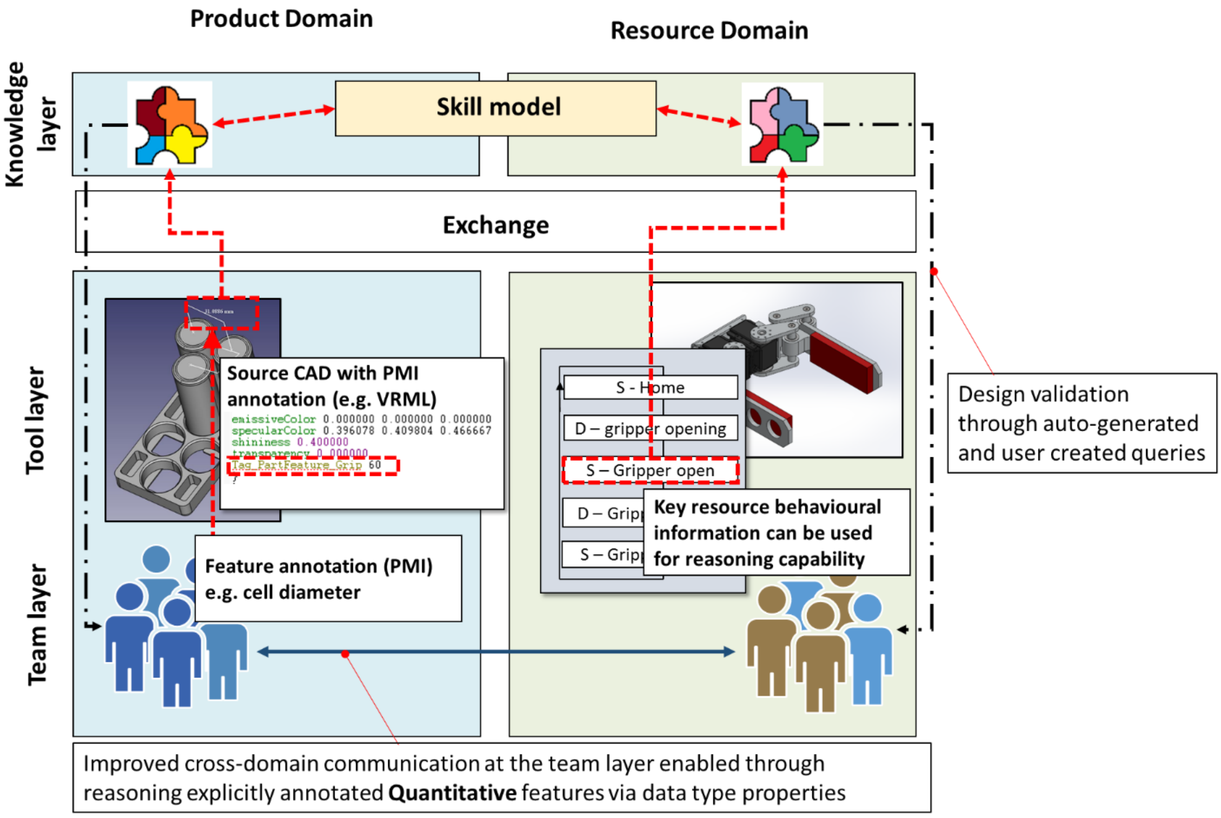
Resource domain. It may well be necessary to include this idea in a future, revised version of the ontology, but it is not within the scope of this work.

Returning to the use of integers as a means for expressing the quantitative information associated with product components, this can be used to check whether a resource is capable of handling the given component. Functions within SPARQL enable the use of simple mathematical calculations and so ultimately a result can be presented to the user highlighting useful information based on the quantitative difference between two values. Depending on the nature of the skill being assessed the result processing will be different. For example, in the case of a weight carrying limit of a machine component, any value of weight of the product less than the limit would result in a positive result. In other cases an assessment would need to be made based on a range. For example, a pneumatic gripper will have an upper and lower value for the size of component it can grip. In this case, provided the product component is within this range, this would produce a positive results. Negative results i.e. indications that the resource bounds are inconsistent with product requirements would highlight how and where changes need to be made either within the Product domain or the Resource domain.

The author has carried out some experiments using Product and Manufacturing Information (PMI) which is supported by several CAD formats (ISO 10303 STEP, ISO 14306:2012 JT) (Chinnathai et al., 2017). PMI is essentially a method to annotate 3D CAD models, usually with geometric dimensioning and tolerance (GD&T) information that has conventionally existed in 2D documents. Maintaining a common model rather than a document through the lifecycle, irrespective of whether this is of the product or the manufacturing system, aligns with the broader model-based, data-driven approach to engineering.

An additional use of PMI is the annotation of key information e.g. annotating the gripping locations of a component. This information can be extracted by parsing the source file and then imported into the relevant data type property of the given component, traceable through unique component identification numbers. As a consequence of this information being present in the ontology, a query can be written to identify whether resources (via the Skill model) are appropriately configured for the product domain’s requirements. Rather than having to process the source CAD for this information, it can be directly gleaned from the ontology as it is explicitly declared. As the ontological model evolves within the business, it is envisioned that annotations of this nature i.e. those associated with process, would become a best practice within industry resulting in the development of standards for how such source CAD should be marked up.

An example of the envisioned workflow described in this paragraph is presented in Figure 3-4. This is taken from a previous work of the author in an attempt to demonstrate how the *QuantitativeFeature* class could be used in a practical way. One of the shortcomings is to have the knowledge that such information can be queried in the first place. In other words, the user may not know that such information exists within the ontology. Although beyond the scope of this work, it is important to also consider how the ontology associated with the ontology is maintained i.e. how to know what is known? This cannot be considered to be a meta-ontology because that would exist at a higher level of abstraction.


Product Domain
Resource Domain
Knowledge layer
Skill model
Exchange
Tool layer
Team layer
21.0886 mm
Source CAD with PMI annotation (e.g. VRML)
S - Home
D – gripper opening
S – Gripper open
D – Grip
S – Grip
Key resource behavioural information can be used for reasoning capability
Design validation through auto-generated and user created queries
Feature annotation (PMI) e.g. cell diameter
Improved cross-domain communication at the team layer enabled through reasoning explicitly annotated **Quantitative** features via data type properties

```
emissiveColor 0.000000 0.000000 0.000000
specularColor 0.396078 0.409804 0.466667
shininess 0.400000
transparency 0.000000
Tag_PartFeature_Grip 60
}
```

Figure 3-4 Workflow diagram showing how data annotated through PMI can enable effective communication and design verification. Particularly within the context of design changes, there is the potential to highlight (almost instantaneously) what aspects of the Resource domain may need modifications and at what level (parameters, logic, structure).

3.4.4 Product Domain summary

The conceptualisation of the ontology for the Product domain as has been described in this section is presented in Figure 3-5 using a UML class diagram. Classes that have ellipses within them represent concepts where the full extent of possibilities have not been conceptualised. This is because it is not necessary to represent all of the possibilities within the scope of this research and also because these concepts have already been well described in existing literature (see Lohse (Lohse, 2006) and Matthew and Rao (Mathew and Rao,

2010) for extensions on the *Liaison* class, and Lanz (Lanz, 2010) and Fenves *et al.* (Fenves *et al.*, 2008) for more general product ontologies). It is important to design the ontology in way that allows extension through clear superclass definitions and by providing examples of sister class concepts. Figure 3-5 further illustrates what exists within the bounds of the Product domain ontology and how it interfaces with the broader PPR model that is presented in Figure 3-2. The reader may note that there is no link between *QuantitativeFeature* and the Skill model as is implied in Figure 3-4. This connection would in fact be managed through queries or rules which would navigate either from *Assembly* or *ProductComponent* (which are connected to the Skill model) to the relevant value. This is elaborated on in the following chapter through case studies.

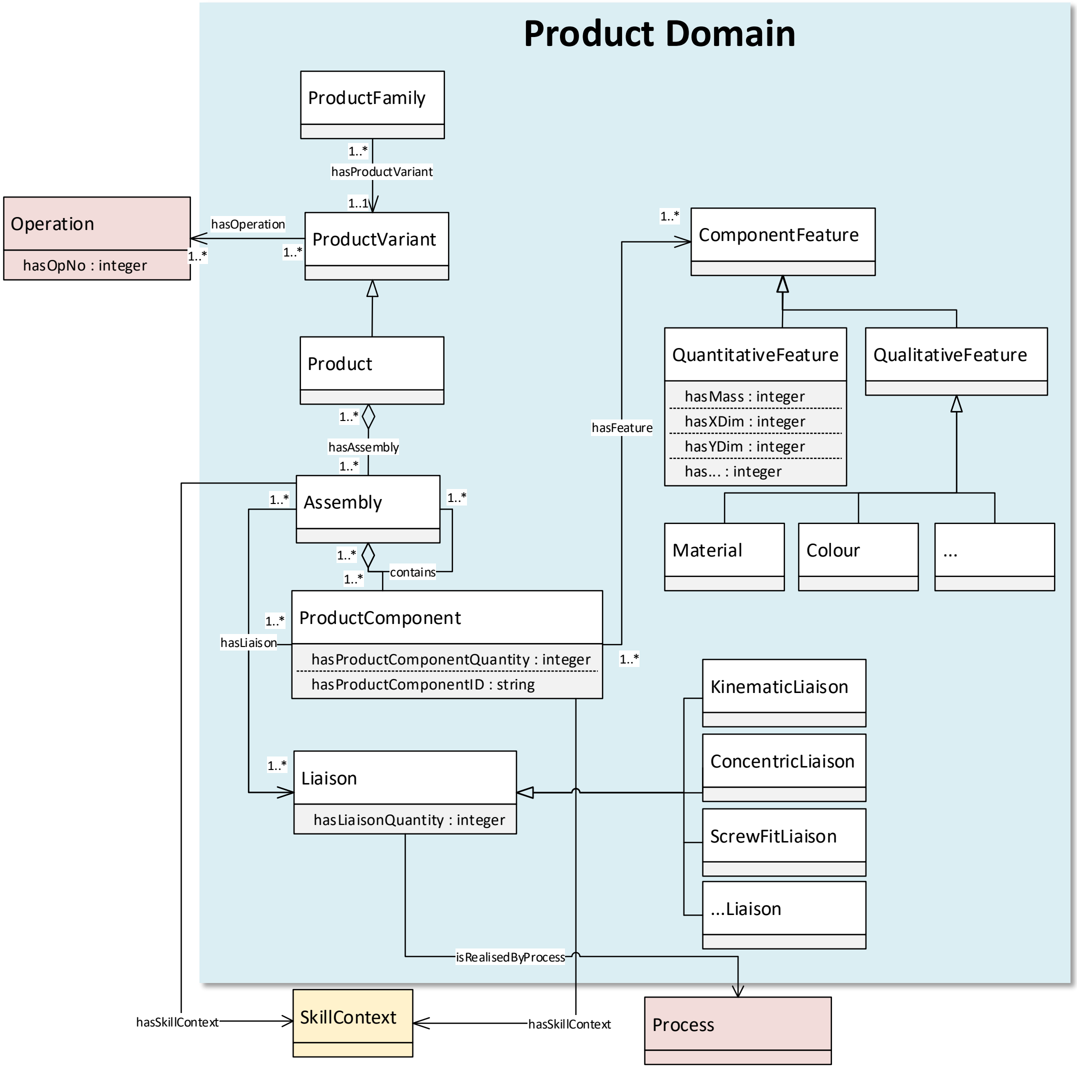


Figure 3-5 Product Domain Ontology

3.5 *Process Domain*

In this section the topology and hierarchy of the Process Domain ontology as well as its connection to the other PPR domains and Skill model is presented. Lohse describes the purpose of the Process Domain ontology eloquently in Lohse (Lohse, 2006) as:

"…the translation of the spatial topological requirements of the product into temporally ordered capability requirements for the assembly system configuration process"

Essentially, this means to transform the Product Domain's undirected graph into a directed one. In Chapter 2, some automated methodologies for achieving this are presented. The role of the Process Domain ontology is not to automate this process however, it is to store the knowledge generated as an output of this reasoning process (be it automated or manual) and link it to the other domains with a view to inferring new knowledge or ensuring consistency between requirements and capabilities. In Figure 3-3 the liaisons that exist between product components have been given numerical names. The numerical values represent unique identification numbers (IDs) which the Process Domain transforms into first a high level sequence and then a more granular description of the activities required to achieve a given liaison.

It is clear that there is a need to define the concepts for the hierarchy in the Process Domain ontology, however in contrast to the unsubstantiated claim of Lohse (Lohse, 2006), there remains (to this day) a lack of convergence on the levels, terms, and even the activities/responsibilities of this domain. Table 3-2 presents an overview of some works that have presented an ontology within the Process Domain. This list does not claim be entirely exhaustive or comprehensive review of Process Domain terminology, largely because search terms are unable to reveal hierarchies that may well use similar concepts through different words. Although not directly relevant, the terminology used in the Microsoft project manager software – Microsoft Project, is also referenced. This is because an analysis of existing process representations highlighted that project management tools such as Gantt charts were a specific type of representation that used their own semantics (Knutilla *et al.*, 1998). However, to the author's knowledge, consideration of the semantics used beyond the domain of manufacturing have not been considered to derive Process Domain ontologies in manufacturing in previous works.

Table 3-2 Summary of Process Domain hierarchies presented in the literature

Author	Hierarchy (high to low)
Lohse (Lohse, 2006)	Activity, Process, Task, Operation, Action
Lanz (Lanz, 2010)	Activity, Process, Task, Operation, Action, Sub-action
Lastra (Lastra, 2004)	Manufacturing Process, Assembly Task, Assembly Process, Assembly Operation
Demoly *et al.* (Demoly *et al.*, 2010)	Assembly Operation, Process
Borgo and Leitão (ADACOR) (Borgo and Leitão, 2007)	Process Plan, Operation
Ramis Ferrer *et al.* (Ramis Ferrer *et al.*, 2016)	Operation, Process, Task
Process Specification Language (PSL) (Bock and Gruninger, 2005, Grüninger, 2004)	Activity, Subactivity, Primitive
Microsoft Project (Chatfield and Johnson, 2010)	Summary Task, Subtask

The conclusion that can be drawn from Table 3-2 is that while some convergence exists with respect to the words, the hierarchical positions in which they appear are not consistent. This does not appear to be the case in the Product or Resource domains where the semantics remain largely consistent, albeit with differing topologies depending on the stance or perspective of the creator. This may be the case because both of these domains exist physically. On the other hand, the Process Domain is inherently abstract in nature. It is perhaps the domain most aligned with the definition presented by Borst *et al.* in (Borst *et al.*, 1997) as the need for explicitly specifying what is only a "shared conceptualisation" is most apparent in the Process domain.

To elaborate, there are typically physical artefacts generated by the activities of humans within the Product and Resource domains. As a results, if humans no longer exist these physical artefacts will continue to exist. On the other hand, the activities or processes associated with realising these artefacts exist in the minds of humans. It is a shared reality that is not tangible. This philosophical stance is important to express in this way because it:

i) identifies why there is a lack of consistency for defining processes in a systematic way i.e. the conceptualisation is not truly shared due to differing perspectives, cultures etc. (Note that this is partly true in the other domains but less prevalent)

ii) highlights that any choice of terms in the Process domain is likely not to be adopted more broadly. It is more important to define the relationships between the words chosen within this domain and others to generate a meaning

Based on this rationale, despite the importance of formal semantics in ontologies, the terms chosen to describe the Process Domain are not an instrumental part of the methodology. The important aspects are, as mentioned above, the definitions which are defined through the relationships that exist within this domain and between others.

3.5.1 Skills in the Process Domain

The concept of Skill has already been mentioned. Although this is elaborated further later in this chapter, it is necessary to begin describing some aspects of the Skill model in this section due to its strong ties with the Process Domain ontology.

In previous works, the Process Domain has been decomposed into types of activities. Furthermore, due to the limited number of engineering methods or tools for describing process in a way that is both human and machine interpretable, the terminology used to describe the contextual aspect of a process remains non-standard. For example previous works (identified in the literature review that address the knowledge capture of the PPR domains) describe the Process Domain from the perspective of the activities that it must execute. Essentially this means nothing more than representing the terminologies and taxonomies from standards (VDI, 1990, DIN, 2003).

The information regarding *how* or on *what* the activities are to be executed remain elusive. In some cases this could be inferred from the respective links to the other PPR domains. However, often these links are not explicitly described. Typically PPR ontologies have a high level link between domains but the relationships are not described using any terminology to support the definition, it is only stated that a link exists. This has the consequence of it not linking the concepts within the Process Domain to the broader engineering workflow. In other words, if a researcher declares a concept in the Process domain ontology e.g. *operation*, there is little to no description of how the instantiation of this information would be populated into the ontological model in the first place. This demonstrates a lack of connectivity with the practical industrial engineering process associated with deriving a process plan (or any other Process domain activity.) Of course at this point it is necessary to elaborate further on what the Process domain activities may consist of. It must be asserted that this is not a comprehensive assessment. The definitions of the Process domain activities vs. the Resource domain activities are fuzzy at best when considered from a practical workflow perspective as opposed to the hard boundaries assigned by academics.

As this thesis is focused on assembly systems only, the Process domain would consist of activities typically defined by "Assembly Process Planning" (APP). In turn, APP consists of assembly sequence planning (ASP) and assembly line balancing (ALB) (Bikas *et al.*, 2016, Wang *et al.*, 2009). In this research activities associated with ALB are not considered and therefore do not form part of the model.

Beyond the commonly cited activities of APP, the author believes that another dimension must be added to APP which is the process description. This would be the process of ascertaining at an appropriate level of granularity how the directed graph derived from the output of ASP would be executed. Depending on the expertise of the process planner they may be able to describe the nature of resources being used. In the case of a new station/line/factory this will be less obvious, on the other hand for a reconfiguration process this would be much clearer. Therefore, at some stage of the assembly system lifecycle the process description will be quite vague, abstract and disconnected from reality although evolving into something more relatable as the system emerges.

During the reconfiguration stages of the system, to accommodate new products or product variants, there will be a much clearer understanding of the resource requirements and capabilities, and this would be reflected in the process plan's process description. Here, the author identifies a challenge, largely because of the lack of engineering tools and methods associated with representing the process description. Typically, these descriptions could be stored within documents such as word processers, spreadsheets, or flow charts.

There are some process planning tools that are described in the literature concerning computer aided process planning (CAPP) however they are typically focused on producing 3D representations of the process for animation and visualisation. Although they are beneficial to the activities of the Process Domain they do not capture the true essence of the process. In other words we, as humans, may be able to read or watch the process being executed, but it is not done in way that is understandable to a machine. This means that although a visualisation may be present, and even this may be transformed into a set of work instructions, this information is not stored in a way that allows some inferences to be made for future process plans through rules. There are no formal semantics used and this information is not transferred to a knowledge base. Furthermore, there is no dissociation between the aforementioned early and late stages of process planning that exists through the lifecycle of the product or system.

The methodology presented in this research addresses the problems highlighted concerning the lack of consideration for semantics in the Process Domain activities by mapping Process Domain descriptions to two aspects of the Skill concept. Firstly, when describing a process, it is obvious that some form of activity will need to be described. This is represented through the aforementioned standards. The granularity of the nature of the activities could be at a high level such as “place component 1 on component 2”, or at a finer level of details such as “determine the location of component 1, grip the edges at position 1 and position 2, lift component 1… etc.” The process description process follows a workflow from a high level conceptual description which is later rationalised into more detailed descriptions. Thus the Process Domain descriptions need to accommodate differing levels of granularity. In the Process Domain model in this research this is achieved through a three tier model consisting of an *Operation* that consists of an aggregation of *Process* which in turn consists of an aggregation of *Task.* This terminology is derived from the automotive industry, particularly Ford Motor Company with whom the research group has a long-standing relationship. As discussed already, the terms used in this domain are less important that the definitions they represent.

Both the *Process* and *Task* concept consist in part of the explicit actions that have already been discussed. In this research, these are named as a *SkillAction*. On the other hand there is a need to explicitly define the context of the *SkillAction* and this is managed by mapping the *Process* and *Task* concepts to the *SkillContext* class. This class consists of all instances of either the Product Domain or Resource Domain that are tangible concepts e.g. a *ProductComponent* instance from the Product Domain. The explicit declaration of contexts is a key novelty in this research as it allows descriptions to evolve through the lifecycle and, provided a history is maintained, common instances or concepts referred to through differing terminologies can be captured and this knowledge exploited.

Consider the three component assembly presented in Figure 3-6 which is a sub-assembly of a fuel cell (see Figure 3-3). The product is represented in a graph format with the liaisons represented as the edges “a” and “b”. Given that the process planner has derived a sequence whereby “a” must precede “b”, it is necessary to then determine how this might be achieved. At the *Process* concept level, not knowing the nature of the Resource Domain the process planner can speculate on the types of actions required and on what those actions act upon i.e. the context. The objective at this stage, when the context of the Process Domain is in its early stages of maturity is to describe the execution of a *Liaison* and reflect upon the type of actions required to realise this. The evolution of abstract ideas from the Process Domain (represented as graphs with increasing levels of detail) are eventually transformed into ideas that become more tangible.

Having described the *Liaison* execution, the process planner while rationalising through the process will realise that there is a need for certain checks, alignment activities etc. These more nuanced activities would fall under the concept of a *Task*. In and of itself a *Task* is unable to realise a *Liaison*. It is some atomic activity that is ultimately achieved through a change in state of a system. In this way, both the *Task* and the *Process* have been defined independent of what the words themselves may mean. In addition to this, the link to contexts and actions retains the knowledge of what these types of activities were trying to achieve. This information would be made available to the Resource Domain stakeholders (as this domain matures) e.g. machine builders, and some joint activity would derive the type of process plan that is commonly represented within CAPP software tools.

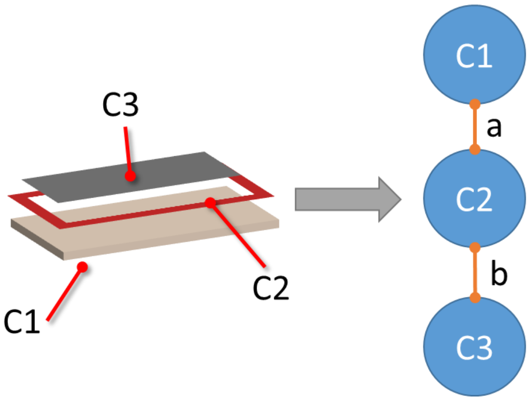


Figure 3-6 Example product demonstrating how the Skill concept is used in the Process Domain

3.5.2 Process Domain summary

The previous sub-section has shed some light as to *why* the author feels it is necessary to link the Process Domain with the Skill model and *how* this has been done. To summarise, this is to follow the workflow that the author recognises is used within industrial environments and captures the knowledge concerning a given process about a given product, within or independent of a context provided by the Resource Domain. This is important as it allows the Process Domain ontology in this research to be more fluid and flexible than existing works have permitted it to be.

To summarise the structure of the Process Domain ontology (illustrated in Figure 3-7), an aggregation of *Task* instances form a *Process* which in turns aggregates to form an *Operation.* An aggregation of *Operation* instances describe a *ProductVariant*. The *Task* is an atomic activity that is executed through the change of state of a machine described in the Resource domain. A *Task* on its own is insufficient to describe the fulfilment of a *Liaison*. The definition of *Process* is therefore that set of *Task* instances that realise a *Liaison*. The *Process* class has a data property value called *hasProcessNo* which provides directionality to a *Liaison* set. The *Task* class also has data property values to describe directionality called *hasTaskNo* which describes the *Task* sequence relative to the *Process.* In order to fully describe a given *Process* or *Task* it is necessary to describe the *SkillAction* and the *SkillContext.*

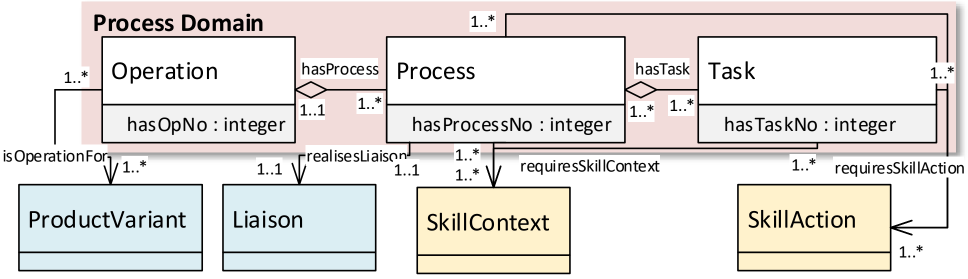


Figure 3-7 Process Domain Ontology

3.6 ***Resource Domain***

Finally, the third domain of the PPR ontology is the Resource domain. This domain in this research follows the structure of a set of virtual engineering tools developed by the Automation Systems Group at the University of Warwick. These tools have been deployed in a number of industrial and research projects to support the lifecycle of a production system from concept generation through to process planning, code generation, virtual commissioning and even supported the operation, maintenance and reconfiguration phases. The Resource Domain ontology has been modelled after these tools because:

i) Accessibility to the data model used in the model allows an accurate ontological model to be created for testing the methodology. Similar engineering tools use proprietary models which cannot be deciphered so readily

ii) The tools have been shown to be of value through the lifecycle and so there is a tangible value for developing the ontological model in line with these tools as a test bed for future industrial research projects for knowledge capture, storage, and inference

This section first describes the engineering tools in more detail. The "as is" data model used in the engineering tools is described with its shortcomings and then the "to be" modelled presented addressing the issues identified. Following this, the Resource Domain ontology is presented which complements the schema of the tools as well as adding some additional functionality.

3.6.1 vueOne engineering tool description

The vueOne engineering toolset capabilities and use within the lifecycle of a manufacturing system are illustrated in Figure 3-8. vueOne is envisioned to be a common engineering environment to support the full set of manufacturing system lifecycle phases enabled by a component-based modelling approach (Lee *et al.*, 2007, Harrison *et al.*, 2016). The tool's extensible data model support process planning, system configuration, code generation and deployment, commissioning, maintenance, operational analytics, and system reconfiguration through different modules. Geometry for system components is converted from native CAD formats to VRML/X3D and form a part of a uniquely identifiable software component. Process planning within the tools is supported through the combination of kinematics and IEC-61131-3 compliant STDs. The tools use a logic engine that interprets the STD to drive the simulation. An XML file that the logic engine uses to drive the simulation can be exported from within the engineering environment and this document is used in this research to connect tool data to the ontological model.

As highlighted in the literature review, commercially available engineering tools with similar capabilities to vueOne are often heavyweight, monolithic, and expensive. Thus, sharing engineering models with the aforementioned stakeholders incur delays and costs that consume valuable engineering time and resources. This is often attributed to complex features, installation procedures, and licensing models. To overcome this, the vueOne viewer is used to share models and simulations at different stage of the development lifecycle to ensure that ideas are being communicated effectively at all levels of the business and through the supply chain. This allows stakeholders to buy into concepts in a more effective way than conventional, fragmented practice, and maintains consistency through the development lifecycle.

At the more granular, detailed engineering of systems, the various components and subsystems are exported from the engineering tools of machine builders into the vueOne engineering tools. The respective model can be added or replaced into the common virtual engineering model (often a crude initial model may be retained as an artefact of the concept development phase) and the associated processes and behaviours are reintroduced. This enables validation of configurations and process plans. In addition, the toolset has the capability to model humans through the *V-Man* (virtual manikin) module and robot behaviour through the *V-Rob* (virtual robot) module. These important elements of a production system can exist within the common model so their interaction can be visualized and assessed to improve and optimize processes and layouts The *V-man* module utilises an intuitive posture manipulation interface and move sequence behaviour is represented through a STD that can be fully integrated to the wider system behaviour through a form of interlock logic. The *V-man* is calibrated through MODular Arrangement of Predetermined Time Standards (MODAPTS) (Carey *et al.*, 2001) which is a type of Predetermined Motion Time System (PMTS) (Harrison *et al.*, 2016). The *V-Rob* module emulates robot behaviour and complements commercial offline programming tools such as ABB's RobotStudio through interfaces to import/export spatial and temporal robot behaviour information.

Retention of domain specific engineering tools negates the need to train engineers on using new tools. Considerably more detailed complementary information exists within such specialist engineering tools, but only what is deemed necessary is brought into the common model. This results in a lightweight model. The common model can then be used later in the lifecycle of the production system to support in virtual commissioning through the vueOne mapper module. This module maps components, PLC function blocks, I/O, and memory addresses, as well as storage and version management of the mapping information.

Beyond the commissioning phase, the lightweight engineering models come into their own as runtime connections through an OPC-UA client that can retrieve data from the physical system and map it to the corresponding virtual component. A standard OPC-UA server is used as it provides access to drivers for a variety of PLCs. This ability to capture runtime data with contextual information is exploited through web-based mobile apps allow monitoring, maintenance, and optimisation with respect to enterprise specific key performance indicators. An overview of the phases of use of the software, its interaction with conventional engineering lifecycle phases for automation systems, and how and where the common model is used as part of CPS are illustrated in Figure 3-8.

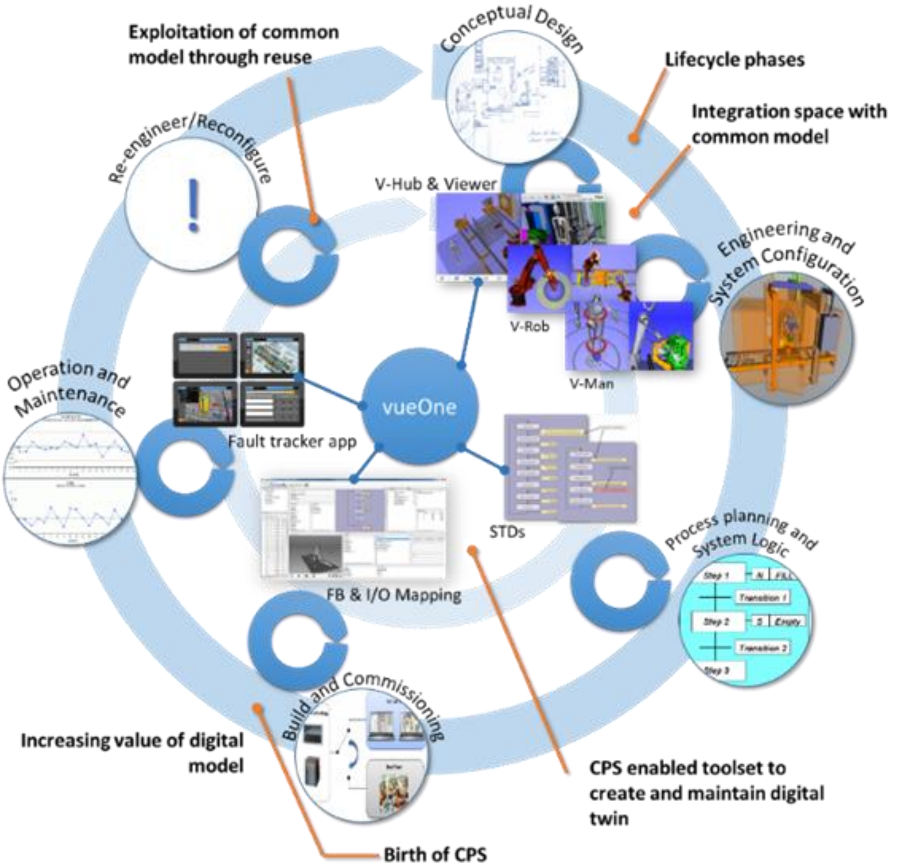

Exploitation of common model through reuse
Conceptual Design
Lifecycle phases
Integration space with common model
Re-engineer/Reconfigure
V-Hub & Viewer
Engineering and System Configuration
V-Rob
V-Man
Operation and Maintenance
vueOne
Fault tracker app
STDs
Process planning and System Logic
FB & I/O Mapping
Build and Commissioning
Increasing value of digital model
CPS enabled toolset to create and maintain digital twin
Birth of CPS

Figure 3-8 System lifecycle supported by the use of a common modelling framework to enable CPS (Harrison;, 2017)

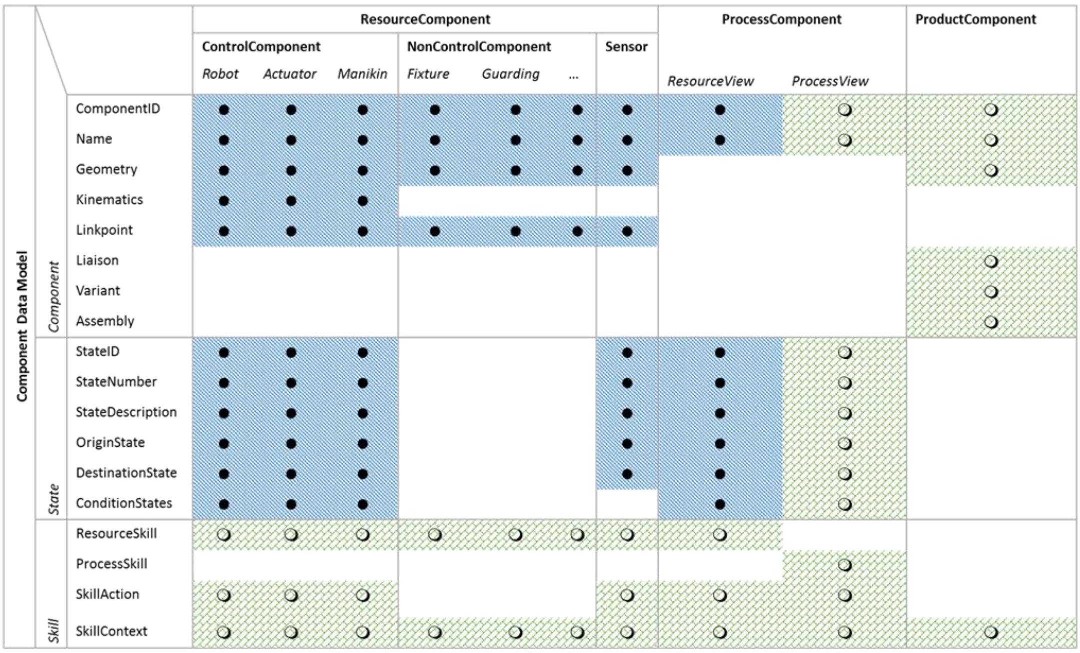
3.6.2 Shortcomings of vueOne and extension

Information concerning the process sequence is stored within a *ProcessComponent* which is a STD with only static states and conditions. The key issue with this approach for executing process planning is that it does not permit a high level view of the process as process steps are already described with respect to resource behaviours. This has a tendency to hide process sequence information that could be related to a directed graph of a product assembly. As a result, the *ProcessComponent* is not particularly accessible to a process planner who may only have a high level view of the process and not details pertaining to the control code. Therefore, in this work, the *ProcessComponent* is modified such that there is a *ResourceView* (the original *ProcessComponent*) and the *ProcessView* which is intended to be used to describe the process at a high level. This would be akin to what one would derive from a directed graph. In order to map the high level *ProcessView* to the low level *ResourceView*, an entirely new set of component properties have been defined that form part of the Skill model.

An additional shortcoming of vueOne is its lack of expressivity with regards to products. Typically, product components are modelled as a *NonControlComponent*. However, this puts them in the same class as objects like fixtures. To address this issue, this research extends the tools to include a specific *ProductComponent* component. This builds upon the work that has been done in (Chinnathai et al., 2017) where the *ProductComponent* was created and then enriched with feature information to enable parametric control logic changes using explicit mappings within a relational database when product component geometrical modifications were made.

Table 3-3 summarises the existing tool data model and the extensions that have been made by the author to enable the approach presented in this research. The reader will note that as the extension is being made to a specific toolset, it could be deemed to be a non-generalizable method. However, many of the concepts that are represented within vueOne are common to industrial engineering tools such as Process Simulate by Siemens. Successful demonstration of the approach within a toolset developed within an academic context can be considered to be a set of recommendations to software developers as to how industrial tools should be extended to enable better integrated data models.

Table 3-3 Existing vueOne engineering tool data model and extensions within this research



Component Data Model		ResourceComponent							ProcessComponent		ProductComponent
		ControlComponent			NonControlComponent			Sensor			
		Robot	*Actuator*	*Manikin*	*Fixture*	*Guarding*	...		*ResourceView*	*ProcessView*	
Component	ComponentID	●	●	●	●	●	●	●	●	○	○
	Name	●	●	●	●	●	●	●	●	○	○
	Geometry	●	●	●	●	●	●	●			○
	Kinematics	●	●	●							
	Linkpoint	●	●	●	●	●	●	●			
	Liaison										○
	Variant										○
	Assembly										○
State	StateID	●	●	●				●	●	○	
	StateNumber	●	●	●				●	●	○	
	StateDescription	●	●	●				●	●	○	
	OriginState	●	●	●				●	●	○	
	DestinationState	●	●	●				●	●	○	
	ConditionStates	●	●	●					●	○	
Skill	ResourceSkill	○	○	○	○	○	○	○	○		
	ProcessSkill									○	
	SkillAction	○	○	○				○	○	○	
	SkillContext	○	○	○	○	○	○	○	○	○	○

Key: ● Existing tool capabilities ○ Tool extension

3.6.3 Resource Domain ontology

The Resource domain ontology is represented in Figure 3-9 and is based on the vueOne engineering tool data model (Table 3-3). However, many concepts and relations are common to existing resource domain ontologies (Lohse, 2006, Lastra, 2004, Järvenpää, 2012, Lanz, 2010) demonstrating the generalisability of the methodology. The highest level concept in this domain is that of *ManufacturingSystem* that is composed of *Station* instances. A *state* is linked to the Skill model through *executesSkillAction*. Instances of *Sensor*, *ControlComponent* and *NonControlComponent* can be linked to the Skill model via *hasSkillContext* as they are objects that exist in the physical world. The lack of a formal constraint applied between these classes and *SkillContext* provides flexibility in how skills are described. In (Lohse, 2006) a similar differentiation is made between what the authors' define as *ControlComponent* and *NonControlComponent* through concepts called Active and Passive using an FBS approach. In addition to describing logic, the *State* class in conjunction with the *ElementType* class support in the selection of instances of *FunctionBlock*. This is an extension to the work presented in (Ramis Ferrer *et al.*, 2015a).

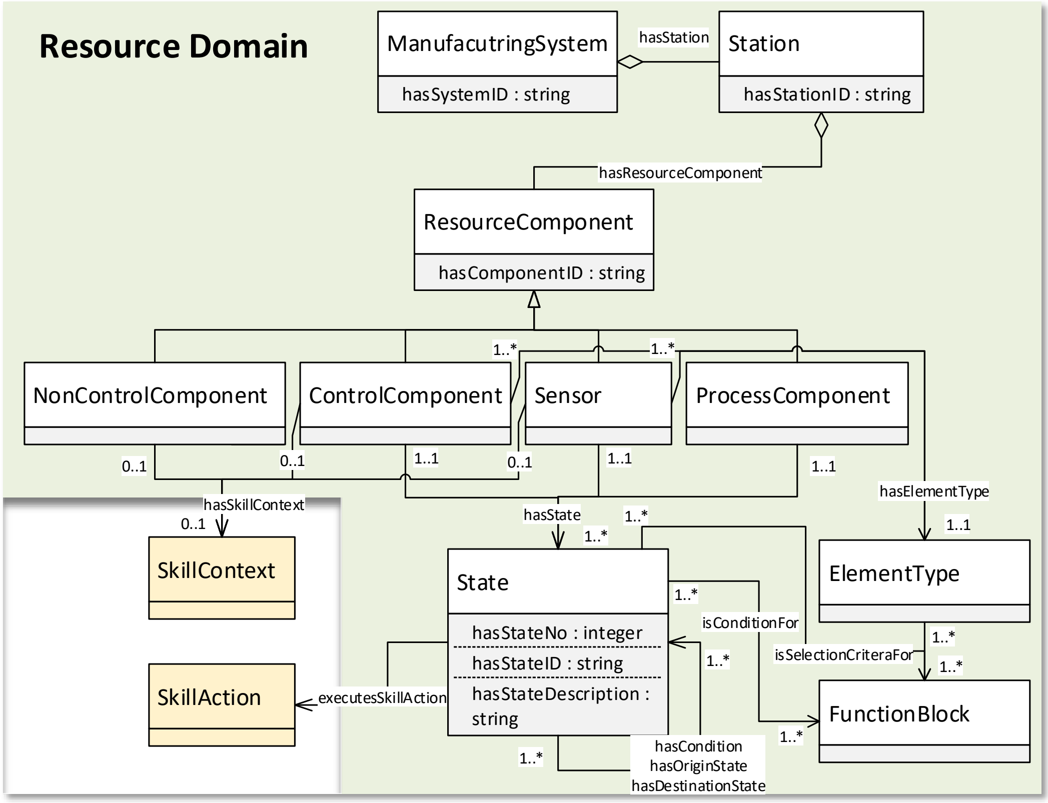


Figure 3-9 Resource Domain Ontology

3.7 *Skill model*

3.7.1 Rationale for terminology

The objective of the Skill model is to bring together the PPR ontologies to negate the need for explicit mapping between them as far as necessary by inferring where connections or links should be made. The Skill model should be able to describe what the Product Domain and Process Domain require while describing what the Resource Domain is able to do.

Within the context of this research, it was necessary to pin down which term to use and appreciate what the connotations of word choice may be should the model be extended going forward. The word chosen for the model needed to be general, so as to meet the above objective while ensuring that there was limited semantic conflict with other concepts (Mens, 2002). As discussed in the literature review in 2.5.3, there are a number of terms that are used by academics to describe what a "thing" is able to do. The most common vocabulary used are "Skill" (Pfrommer *et al.*, 2013, Schleipen *et al.*, 2014), "Capability" (Järvenpää, 2012), and "Function" (Lohse, 2006).

The term "Function" is defined by the Oxford English Dictionary as a noun that is "an activity that is natural to or the purpose of a person or thing" and a verb "work or operate in a proper or particular way". There is some level of semantic conflict between the word "Function" and the terminology used within the context of PLC programming when describing one of the five IEC-61131 stipulated languages, namely Function Blocks (which appear in the Resource Domain ontology). To prevent any confusion, the term "Function" was not been chosen.

The term "Capability" is defined by the Oxford English Dictionary as a noun which is have "the power or ability to do something". Within the context of manufacturing, the term "Capability" is also heavily used within the "Six Sigma" paradigm that is embraced by many industries (Pyzdek and Keller, 2014). There are a number of statistical measures used that fit within capability studies. These measures help to identify whether a process can meet customer requirements. Due to the use of this word in this area and its prevalence, it is likely that should the model be extended, it would be valuable to know not only whether a given resource is able to execute something, but how well or how *capable* it is at doing so.

The term "Skill" has therefore been chosen as i) it has been chosen by more modelers than other terms within this context in the literature and therefore lends itself better to integration should models be brought together, and ii) is less likely to run into semantic conflict if the model is extended as the author has not found examples of this term being used in models or tools beyond this context.

3.7.2 Skill model description

Each of the respective PPR domain models contain some link to either the *SkillContext* class or the *SkillAction* class. The former is defined as a noun which is often either a location or an object. The latter is defined as a verb and a standardised set could be used from existing standards such as VDI 2860 or DIN 8580. Alternatively, the taxonomy provided by (Järvenpää, 2012), (2008) or (Huckaby and Christensen, 2012) would also be a suitable approach. The aggregation of a *SkillContext* and *SkillAction* form a *ProcessSkill* or *ResourceSkill* depending on the source domain of the context and action. A *SkillContext hasEquivalentSkillContext* with an instance of *SkillContext*. This is to allow equivalent concepts to be linked so that descriptions related to the same thing, but described using domain specific languages can be mapped. This addresses, in part, the issue of multiple aliases for a given entity discussed earlier in this section. The dashed line in Figure 3-10 is an inference based on the mappings of *ProcessSkill* and *ResourceSkill* according to their respective relationships with *SkillAction* and *SkillContext* instances.

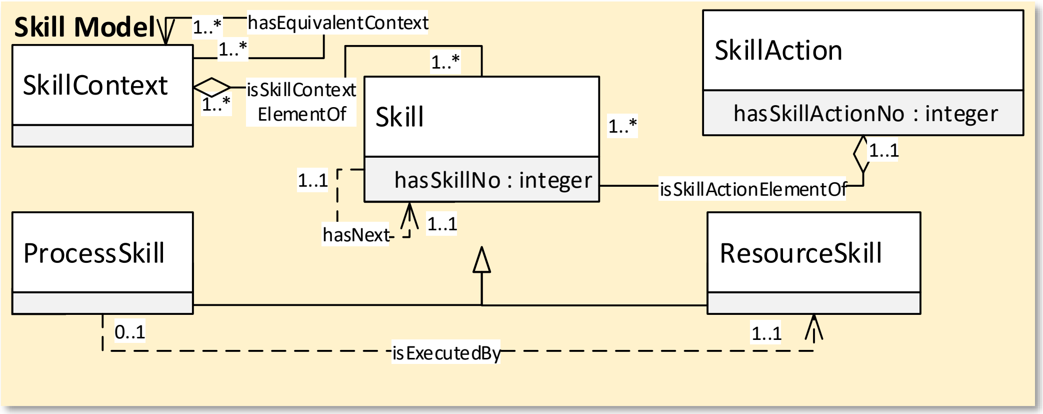


Figure 3-10 Skill Model

3.7.3 Model Enrichment

According to the W3C OWL Reference[2], the properties in OWL have a direction from domain to range, however to facilitate full and flexible navigation, it is of benefit to define relations in both directions. OWL has built-in property called *"inverseOf"* that reduces the

[2] https://www.w3.org/TR/owl-ref/

manual effort and thus risk of error when implementing reverse object properties. Table 3-4 describes the use of some of the *inverseOf* object properties added to the model.

Table 3-4 Excerpt of addition of "inverseOf" object properties

Domain	Object Property	Range	inverseOf Object Property
State	executesSkillAction	SkillAction	isExecutedBySkillAction
Operation	hasProcess	Process	isProcessOf
Process	hasTask	Task	isTaskOf
Station	performsOperation	Operation	isPerformedBy

3.7.3.1 Rules

Using the Semantic Web Rule Language (SWRL) a number of rules are implemented that provide a degree of high level consistency across different concepts. These are presented in Table 3-5. Rule 1 facilitates the mapping of *ProcessView* skill requirements with *ResourceView* skill capabilities. Note that instances of *ProcessView* are within the Process Domain ontology within the *Process* concept. On the other hand the respective *ResourceComponent* subclasses that have states within the Resource Domain ontology are instances for the *ResourceView.* Rule 2 is used as the *inverseOf* object property cannot be used on the *hasState* object property. This is because the *ProcessComponent* as well as the *ControlComponent* and *Sensor* also use the *hasState* object property. Thus, an *inverseOf* approach for adding bi-directionality would result in the reasoner incorrectly inferring that the *inverseOf* a *state* of *ProcessComponent* are also states of *ControlComponent* and *Sensor*. Rule 2 was implemented to allow inferences to be made as to what instances of *ElementType* or *ControlComponent* the *FunctionBlock* class can be used for using the *canBeUsedFor* object property. In order to depict how Rule 2 works, is presented below. The solid line indicates the explicit mapping as already presented in Figure 3-9, while the dashed line represents the new link as a result of Rule 2. Therefore, this rule permits the semantic reasoner to infer which function block can be used for which instances of *ControlComponent* and/or *ElementType* (Ramis Ferrer *et al.*, 2015b).

Table 3-5 Addition of SWRL Rules

Rule name	SWRL syntax
Rule 1	ResourceSkill(?a) ^ isSkillContextElementOf(?d, ?b) ^ isSkillContextElementOf(?d, ?a) ^ ProcessSkill(?b) ^ isSkillActionElementOf(?c, ?b) ^ isSkillActionElementOf(?c, ?a) -> isExecutedBy(?b, ?a)
Rule 2	State(?s) ^ ProcessComponent(?p) ^ hasState(?p, ?s) -> isStateOfProcessComponent(?s, ?p)
Rule 3	ControlComponent(?x) ^ ElementType(?z) ^ hasElementType(?x, ?z) ^ FunctionBlock(?f) ^ isSelectionCriteriaFor(?z, ?f) -> canBeUsedFor(?f, ?z) ^ canBeUsedFor(?f, ?x) ^ canUseFunctionBlock(?x, ?f)

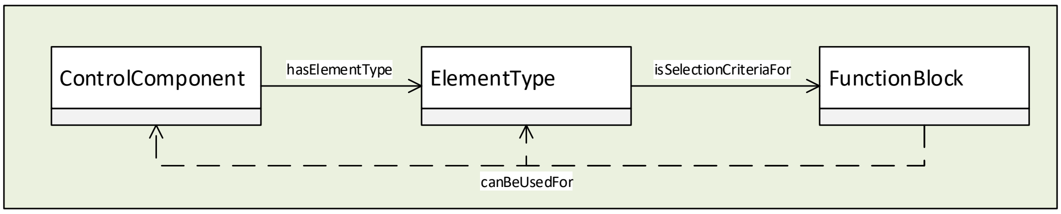


Figure 3-11 Illustration of Rule 2

3.8 *Inconsistency management*

One of the key reason for using ontologies is their ability to reason and thus maintain consistency. In this section, the way this reasoning power is used to support the engineering workflow is described. In the following chapter, the methods used are instantiated with some case examples.

3.8.1 Capability checking

It should be noted that the knowledge base created by instantiating the ontological model forms a backend to vueOne with knowledge transfer within and across domains as illustrated in Figure 3-1. Within an engineering context, knowledge is usually transferred between respective domain stakeholders through documents that are either physical or digital in nature, or via digital models. These exchanges of information include large amounts of data and although there is a degree of common understanding as to what a given document/model could/should contain, there remains an exercise of consultation to extract the relevant information. To address this, this research proposes the use of SPARQL queries that are able to infer implicit knowledge from explicit links and rules. The focus of these queries is to allow the extraction of information from the Resource domain as this is the most complex, with the longest lifecycle and highest re-engineering costs. It is reasoned that if more information is made available with regards to its capabilities when introducing new products or product variants, then the engineering process can become more streamlined as changes to the product or system (whichever is deemed to be a priority by respective stakeholders and the business more broadly) can be made sooner and with a clear direction.

Figure 3-12 presents a workflow example to appreciate how the ontological domain models in conjunction with the queries would interact with the decisional workflow and thus put the presented work into context. The structure aligns with the vision presented in Figure 3-1 and thus manages in part the “human interaction” aspect which is often error-prone. Query 1 focuses on whether or how the physical requirements e.g. mass, weight, physical features, of the Product domain can be determined to be fulfilled by the Resource domain. Information outputted from Query 1 would support in the determination of whether product requirements are met by Resource domain skills/capabilities.

New market demands, often driven by externalities, would result in the development or modifications of existing product lines. Product designers synthesize these requirements and generate modified CAD models which in turn generate a corresponding bill of materials (BOM). The CAD models would be parsed for the relevant information and updated. The links from the Product Domain to the Resource Domain largely deal with aspects that would influence the mechanical design of shop floor equipment. The output of Query 1 would therefore ascertain whether the mechanical requirements have been fulfilled and the results of the query would be passed back to the product designers and the responsible mechanical engineering team. Based on the output of this query the relevant stakeholders would know what the impact of the modified product is and what changes, if any, need to be made early in the change propagation lifecycle.

On the other hand, Query 2 focuses on addressing the question as to how the requirements of the Process domain can be determined to be fulfilled by the Resource domain component logic as opposed to general machine capability. As a consequence of the product design change, the process planner would need to determine what changes to the process need to be made and, in turn, begin modifying the bill of process (BOP) and the assembly process plan (APP). Note that this activity can be automated as has been demonstrated in (Pintzos *et al.*, 2016). The modifications to the respective process documentation would update the relevant classes in the Process domain ontology. This would in turn trigger Query 2 and advise whether the modified process requirements are executable by the relevant shop floor equipment and what changes, if any, need to be made. This information would be passed onto the relevant process planner as well as the responsible machine software programmer who could collaborate and work towards finding a solution. As the Resource domain is linked to the Product domain and Process domain in the reverse direction also, any changes would be highlighted to the stakeholders of the respective domains allowing full transparency of the system's state.

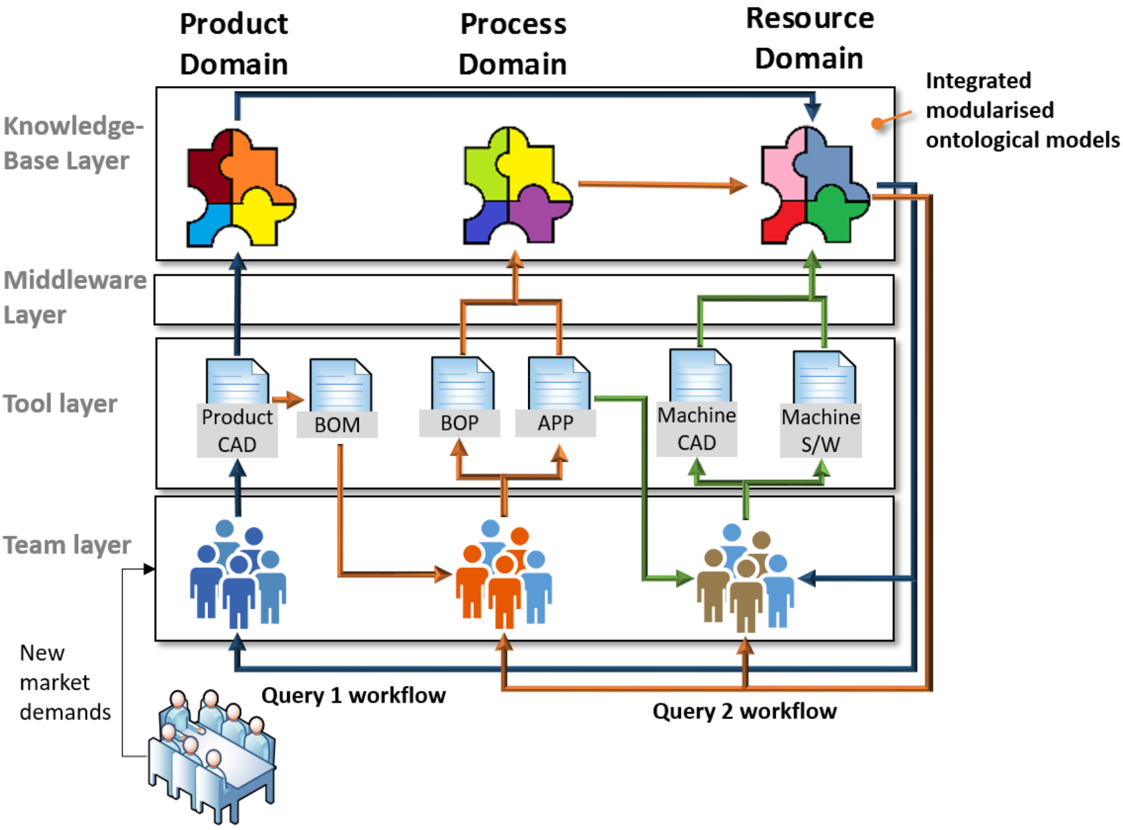


Figure 3-12 Example of how ontologies and queries are used to support engineering and decisional workflow

Having described the basis for Query 1 and Query 2, it is necessary to consider how they can be created in a generalized way by examining the ontological structure and determining the best "route" that the respective queries should navigate.

For Query 1, questions of a mechanical nature arise such as "is the fixture large enough for my part?" and "can the robot lift the part?" etc. At the highest level this can be addressed by the *SkillContext* classes. The arrows in red with the small dashes in Figure 3-13 illustrate how the mappings to this class allow a comparison to be made without have to explicitly map between the Product and the Resource domain. When the users of the ontology interact with the front end tools, the author proposes that it would be incumbent on them to define the context of the engineering work they are doing within the broader context using the library of information that exists within the ontology. This would make use of the *hasEquivalnetSkillContext* relationship which would then allow the person within the Product domain querying the system, to quickly identify what within the manufacturing system is relevant to their *ProductComponent.* For example, if the product designer wishes to identify whether a fixture has the correct dimensions for a new *ProductComponent* they could query, via the *SkillContext*, what fixture *hasEquivlaentSkillContext* with the *ProductComponent*. From this, the product designer could directly interrogate the CAD model or discuss with the relevant stakeholder in that domain to understand what could be done to make changes for the new *ProductComponent*. This is an example of how explicit knowledge capture from humans that are interacting with the ontological models, via engineering tools, can be used through lifecycle phases. As the ontology evolves, certain attributes about certain components, both within the Product and the Resource domain, would become a permanent property that is always queried. Knowing this, a datatype property can be added to the relevant classes to accommodate this knowledge e.g. a specific dimension of a *ProductComponent* that varies with application that aligns with a specific dimension of a *Fixture*.

More detailed queries could then be written which then automate the aforementioned manual interrogation process. One of the benefits of ontologies is the ability to query the model itself and not just the instances. Therefore, if a new user arrives into the organisation and is not aware of the relation between these specific dimensions, they could query the nature of the knowledge that already exists within the system resulting in an understanding what information can be queried in the future. Based on this high level capability checking between the Product domain and the Resource domain, a query using SPARQL syntax is presented in Figure 3-14**.**

For Query 2, Figure 3-13 uses solid red arrows to indicate the expected route that would need to be navigated to check that the Resource domain was capable of executing the requirements of the Process domain. In order to ascertain that the full set of Process domain requirements are being met, it is necessary to decompose down to the *Task* level as it is at this level of granularity that a relationship exists with the Resource domain via the Skill model. Encapsulation of states to aggregate a *Skill* into a more complex *Skill* so that it is something that be mapped directly to a *Process* is possible, but not addressed in this research. This would be akin to the vision of IEC-61499 (Vyatkin, 2009) or the work of Wang *et al.* (Wang *et al.*, 2012, Wang *et al.*, 2008) although with the additional power of reasoning. Based on the aforementioned workflow, whereby the user defines *SkillAction* and *SkillContext* for their respective domains, through reasoning based on Rule 1 (Table

3-5), a mapping would be created between *ProcessSkill* and *ResourceSkill* (Figure 3-10). This would advise the user, who is likely to be the process planner in this case, as to whether the process that has been designed is executable. This is based on the *Skill* of the *State* of a *ControlComponent*. If findings are made to the contrary based on the query, an engineering change workflow can be set in motion to modify the control logic of the machine. The checking of capability execution between the Process domain and the Resource domain is presented as a generalised SPARQL query in Figure 3-15**.**

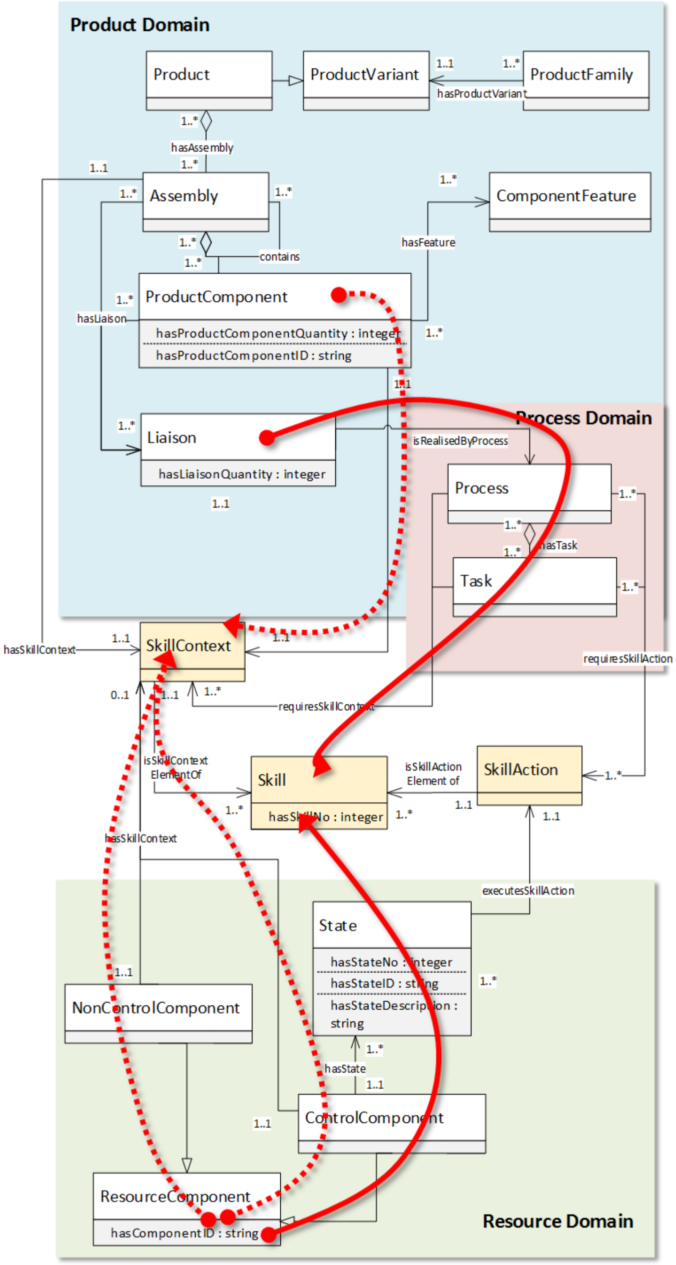


Figure 3-13 Ontological model navigation for capability checking

```
PREFIX ProductDomainOntology: <httpy//www[…]ProductDomainOntology.owl#>
PREFIX ResourceDomainOntology: <httpy//www[…]ResourceDomainOntology.owl#>
PREFIX SkillModel: <httpy//www[…]SkillModel.owl#>

SELECT ?ProductComponent ?NonControlComponent ?ControlComponent

WHERE {

?Assembly ProductDomainOntology: contains ?ProductComponent
?ProductComponent PPRSkill: hasSkillContext ?SkillContext
?NonControlComponent PPRSkill: hasSkillContext ?SkillContext
?ControlComponent PPRSkill: hasSkillContext ?SkillContext
?SkillContext SkillModel: hasEquivalentSkillContext ?SkillContext
FILTER (?ProductComponent = ProductDomainOntology: "an instance of ProductComponent")

}
```

Result

ProductComponent	ControlComponent	NonControlComponent
an instance of *ProductComponent*	A list of *ControlComponents* that *hasEquivalentSkillContext* as an instance of *ProductComponent*	A list of *NonControlComponents* that *hasEquivalentSkillContext* as an instance of *ProductComponent*

Figure 3-14 Example of how Query 1 could be written for any given instance of ProductComponent using SPARQL syntax

```
PREFIX ProductDomainOntology: <httpy//www[…]ProductDomainOntology.owl#>
PREFIX ProcessDomainOntology: <httpy//www[…]ProcessDomainOntology.owl#>
PREFIX ResourceDomainOntology: <httpy//www[…]ResourceDomainOntology.owl#>
PREFIX SkillModel: <httpy//www[…]SkillModel.owl#>

SELECT ?Liaison ?Process ?Task ?State

WHERE {

?Assembly ProductDomainOntology: hasLiaison ?Liaison
?Liaison PPRSkill: isRealisedByProcess ?Process
?Process ProcessDomainOntology: hasTask ?Task
?Task PPRSkill: requiresSkill ?Skill
?ControlComponent ResourceDomainOntology: hasState ?State
?State PPRSkill: executesSkill ?Skill
FILTER (?Liaison = ProductDomainOntology: "an instance of Liaison")
OR

FILTER (?Process = ProcessDomainOntology: "an instance of Process")

}
```

Result

Liaison	Process	Task	State
an instance of *Liaison*	an instance of *Process*	A list of *Tasks* that aggregate to form an instance of *Process* or *Liaison*	A list of *States* that meet the *Skill* requirements of *Task* to realise an instance of *Process* or *Liaison*

Figure 3-15 Example of how Query 2 could be written for any given instance of Liaison or Process using SPARQL syntax

Making changes of a physical nature e.g. creating a new fixture to meet new product requirements, exist primarily in the physical. Although the shift towards Industry 4.0 could see a CPS approach to physical system changes, there are a number of challenges that have yet to be overcome, primarily the full realisation of Koren's criteria for what denotes a reconfigurable system (Koren and Shpitalni, 2010). On the other hand, due to the abstract nature of software, changes can be made more readily. Furthermore, two types of changes can be made at the software level. The first is the most simple software change that is typically referred to as parametric, whereby a parameter or variable is modified. The next level of change is at the logic level and this is associated with modifying the logic of the machine. Changes at this level could include adding, removing, or swapping states or conditions for a sequence of tasks. The remainder of this section presents a methodology that firstly addresses how the ontology can be used to ascertain an inconsistency between the description of machine control and process requirements from the output of Query 2. Then an approach is presented for how the Skill model can be further exploited to enable the modification of control logic.

3.8.2 Inconsistency checking method

As a consequence of a modified process plan, a machine's control software will also need modifications. These changes could include new process parameters such as magnitude of motion or speed, or logical changes such as sequence. Some aspects of the machine's sequence are linked to mechanical constraints i.e. preventing clashes or ensuring that actuators return to the home position, while other aspects are more functional in nature in that they are directly linked with realising a product requirement i.e. a pick and place operation to fulfil a liaison. This means that some aspects of machine control logic are not mappable to APP as they add no value to the product. This section of the approach focuses on how the output of information from APP activities (considered to be within the Process domain) can be checked for consistency with machine software as an exploitation of the integration that has been achieved through the Skill model. Figure 3-16 illustrates how checks for consistency are made in real industrial environments i.e. through manual interrogation and comparison of documents (Winkler *et al.*, 2016, Lee *et al.*, 2011a, Demoly *et al.*, 2013). The focus of this section and the proceeding one is highlighted by the red connecting line.

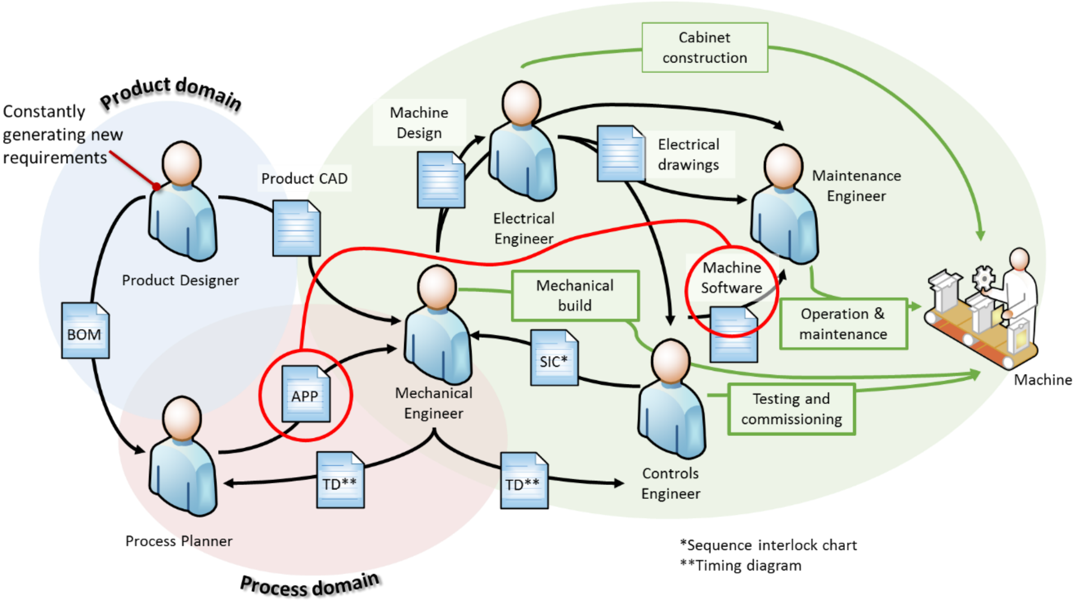
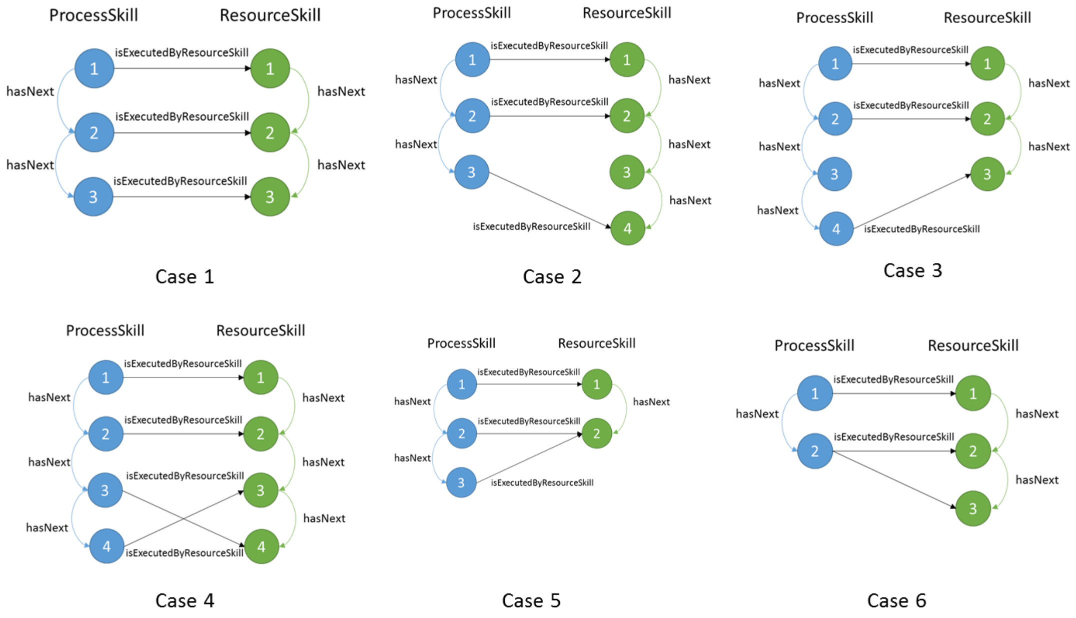


Figure 3-16 Current approach for automation system engineering highlighting focus of this work

To support in the inconsistency management process, the mapping generated by Rule 1 in Table 3-5 is used. In addition the *hasSkillNo* datatype property is a key tool for identifying inconsistencies. For a *ProcessSkill* the *hasSkillNo* value is derived from the *hasSkillActionNo* datatype property which would be declared explicitly by the user during the process of decomposition. On the other hand, for the *ResourceSkill* the *hasStateNumber* value is used. This value is derived from the STD that describes the *ResourceView ProcessComponent.*

Thus, when Rule 1 is implemented and a mapping is created by the *ProcessSkill* and the *ResourceSkill* it is possible to compare the integers associated with them to ascertain inconsistency. This is because there are a finite set of potential mappings that could exist as a result of inferences reasoned from Rule 1. This set of mappings is illustrated in Figure 3-17. Case 1 is the simplest of all cases and unlikely to exist in reality because *ResourceSkill* instances will describe steps not considered in *ProcessSkill*. Regardless, no inconsistency is identified here. Case 2 is expected to be the most common case whereby the description of machine behaviour has a greater degree of granularity and therefore there exist steps that are not considered by the Process domain. In this case, the numerical value associated with *ProcessSkill hasSkillNo* will always be less than or equal to the value associated with *ResourceSkill hasSkillNo*. In Case 3 there is an instance of *ProcessSkill* that is unmapped to *ResourseSkill*. This indicates an inconsistency in that all of the requirements of the Process domain have not been met. The integer value of *ProcessSkill* that is mapped post the unmapped instance has a greater magnitude than its corresponding *ResourceSkill*. Case 4 is an example of where the mapping between the respective skills has been flipped. The link from Step 3 to Step 4 from the *ProcessSkill* to the *ResourceSkill* in and of itself does allow one to determine whether or not the respective descriptions are inconsistent. This is because this scenario is identical to Case 2. However, the link from Step 4 to Step 3 from the *ProcessSkill* to the *ResourceSkill* denotes the inconsistency. This is due to the assumption that the *ResourceSkill* description is at least as detailed as the *ProcessSkill* description. If this assumption is true, then a larger integer being mapped to a smaller integer from the *ProcessSkill* to the *ResourceSkill* instantly denotes an inconsistency. Case 5 and Case 6 denote a many-to-one relationship. Both are examples which should not be possible due to the common atomic methods for describing the *ProcessSkill* and the *ResourceSkill*. If this does arise it indicates that the method has been used incorrectly and the descriptions should be revisited for the given steps.



Case 1 Case 2 Case 3

Case 4 Case 5 Case 6

Figure 3-17 Mappings between ProcessSkill and ResourceSkill as an outcome of Rule 1 to use as a basis for consistency checking

From the case-based analysis it is concluded that a sequence inconsistency exists if the result from subtracting the integer associated with *ResourceSkill* from the integer associated with *ProcessSkill* is positive. Due to the limitations of the ontology editor used in this work (Protégé) the inferences generated by the reasoner through the SWRL rule cannot be queried and thus exploited. Therefore, a SPARQL query (Figure 3-18) is written which replicates the inferences generated and then finds the difference between the integers associated with the respective skills. This is a general query that can be used in any use case that utilises the Skill model used in this work.

```
PREFIX SkillModel: <httpy//www[…]SkillModel.owl#>

SELECT DISTINCT ?ProcessSkill ?ProcessSkillNo ?ResourceSkill ?ResourceSkillNo ?InconsistencyCheck

WHERE {

?ProcessSkill rdf:type SkillModel:ProcessSkill.
?ResourceSkill rdf:type SkillModel:ResourceSkill.
?SkillContext SkillModel:isSkillContextElementOf  ?ProcessSkill.
?SkillAction SkillModel:isSkillActionElementOf  ?ProcessSkill.
?SkillContext SkillModel:isSkillContextElementOf  ?ResourceSkill.
?SkillAction SkillModel:isSkillActionElementOf  ?ResourceSkill.
?ProcessSkill SkillModel:hasSkillNo ?ProcessSkillNo.
?ResourceSkill SkillModel:hasSkillNo ?ResourceSkillNo.
BIND (?ProcessSkillNo - ?ResourceSkillNo as ?InconsistencyCheck)
} ORDER BY ASC (?ResourceSkillNo)
```

Result

ProcessSkill	ProcessSkillNo	ResourceSkill	ResourceSkillNo	InconsistencyCheck
A set of instances of all *ProcessSkills*	a list of *ProcessSkillNo* datatype properties associated with the corresponding *ProcessSkill*	A set of instances of all *ResourceSkills*	a list of *ResourceSkillNo* datatype properties associated with the corresponding *ResourceSkill*	Numerical difference between *ProcessSkillNo* and *ResourceSkillNo*

Figure 3-18 Inconsistency check query using SPARQL

3.8.3 Modification of logical changes through virtual engineering and ontologies

Having identified whether or not an inconsistency exists between APP descriptions and machine control software in section 8.2, this section addresses how such an inconsistency can be resolved. The SPARQL query that has already been described in Figure 3-18 is used as an indicator of an inconsistency. One the shortcomings of using SPARQL within the Protégé environment is the inability to exploit the resulting data outside of the ontological model. Therefore, this deviates from the original vision that is illustrated in Figure 3-1. To address this, a framework which extends what has already been alluded to in Figure 3-12 is illustrated in Figure 3-19 and described as follows.

The vueOne toolset is able to export the logic associated with a simulation in an XML format. The General Architecture for Text Engineering (GATE) is used to semantically annotate the exported XML so that the data can be automatically instantiated within the ontology. GATE cannot be considered truly within the semantic web technology family, but classed more generally as a semantic technology (2010). GATE's primary function is that of text analysis with components for parsers, morphology, and information extraction, among others. GATE is implemented within a Java component model. With the release of GATE 3.1, support for ontologies was added which are classified as language resources within the GATE framework. In this work the Ontology Annotation Tool (OAT) is used that is available from the broader Ontology Tools plugin set. Within this environment, the user can manually annotate a source file with respect to one or more ontologies. This allowed the authors to link tags from the engineering tool XML to the appropriate ontology class as well as explicitly define relations that are not always clear from the source XML. The Java Annotation Pattern Engine (JAPE) is then used to populate the ontology. It should be noted that implementation beyond the vueOne engineering toolset presented in this work would require an exercise in collating the multiple aliases that may exist for a single entity. This process of name normalisation would be supplemented with knowledge concerning the naming convention of common concepts in other engineering tools so that, regardless of the source file's textual form, a given entity is linked to the same ontology instance.

In order to manipulate the ontology the Apache Jena framework is used which provides a greater degree of flexibility as compared to the tools available within the Protégé environment. Apache Jena is an open source Java framework to support in the development of Semantic Web applications. It provides an API to manipulate RDF triples, supports OWL, the execution of SPARQL queries, as well as a rule-based inference engine. Although ontology editors such as Protégé share some of the functionality of Jena, due to the latter being implemented within Java, GUIs that are user friendly, intuitive and do not require expert knowledge to operate can be developed. In other words, Jena has the capability for developers to create a front end for end users within industrial environments to use, while Protégé can usually only be operated by experts. In addition, Jena is scalable and provides the most complete, easy to use framework as compared to its competitors e.g. Sesame (Jaiswal *et al.*, 2015). Furthermore, its flexibility accommodates the implementation of a tailored solution allowing the authors to develop bespoke logic resulting in a powerful decision support tool.

The manipulation of the ontology is executed by the user, who is most likely to be the process planner, through a Java graphical user interface (GUI). The input data from the user updates the process sequence at a high level and, due to the reasoning power of ontologies, updates are made to the low level control logic. This in turn generates an updated version of the XML file which is interpretable by vueOne for visualisation of the machine based on the new process. Once the process has been validated it can be shared with a controls engineer for approval. The updated process can be transformed into PLC code and deployed to the physical machine. Dashed arrows in Figure 3-19 represent information or knowledge that are artefacts of upstream lifecycle phases such as initial design, engineering etc. Some of the steps illustrated such as auto-code generation (denoted by the link from vueOne to machine s/w), have already been developed (Harrison *et al.*, 2016) and are thus not within the scope of this work.

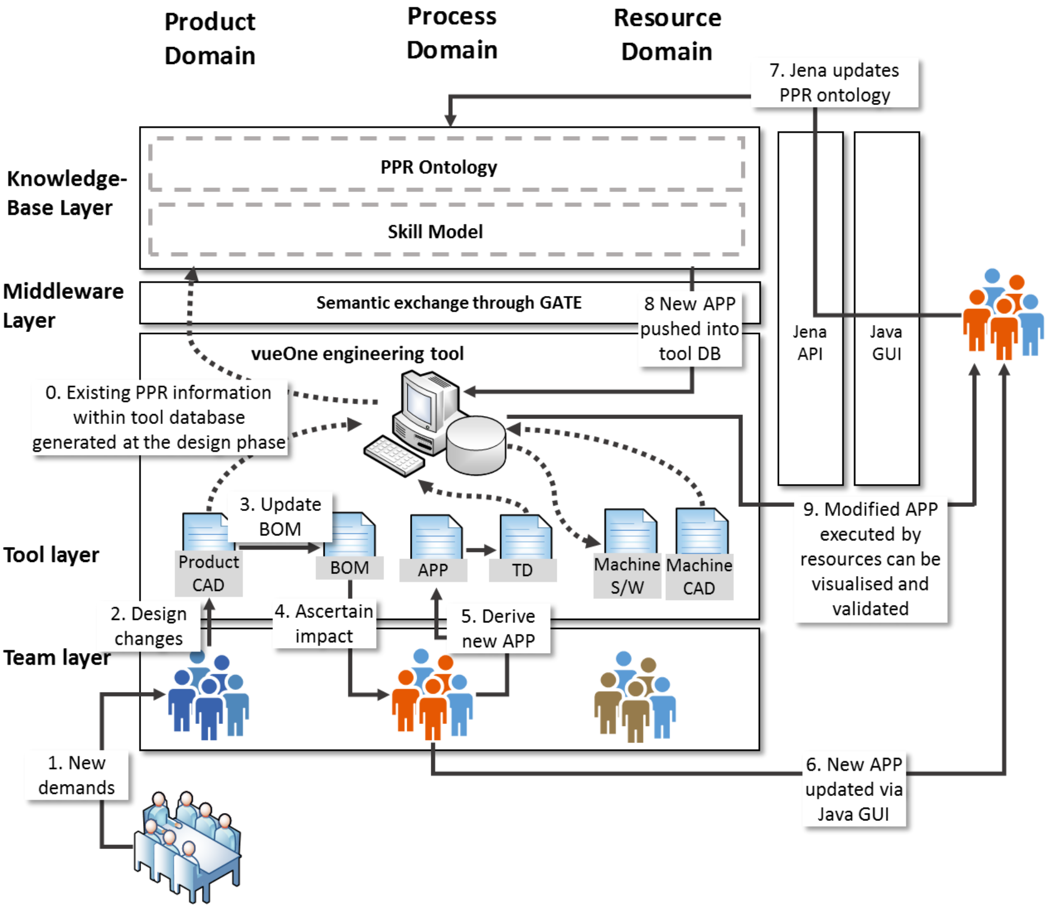


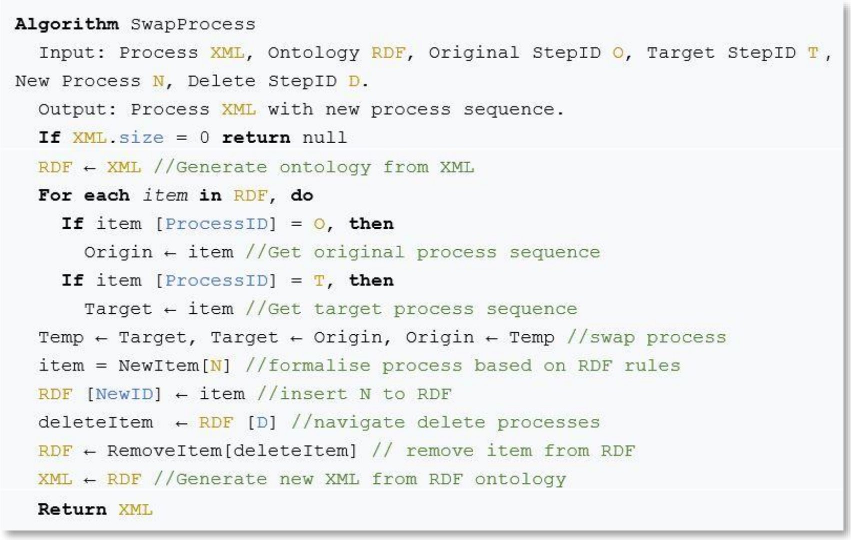
Figure 3-19 Full framework to realise vision

3.8.3.1 Algorithm for executing changes

Having described how the high level process plan is connected to the low level machine control code in previous sections, the next step is to present how this mapping is exploited to enable changes. Figure 3-21 presents some pseudo code that describes the algorithm for swapping, adding, or deleting steps. The input for the algorithm, which is implemented within Java using the Jena framework, is the source XML from the engineering tools denoted as the “Process XML” in the code below. In addition, the PPR ontology together with the Skill model is required as the rules within this model allows the changes to be made in a consistent way. Through GATE, the XML is auto-instantiated into the ontology so that the latest version of the system is available. Via the GUI the user has three options. Firstly, it is possible to swap a step. The method for doing so requires the user to indicate the original process step and the target step.

Through the connections made via the Skill model the system is able to identify the relevant control logic states that are associated with the process step in the *ProcessView* and swap them accordingly. Note that when swapping steps, conditions remain in their original location to allow the code to be executed. Next it is possible to add a step provided that the system is capable of doing so. When introducing a new step, a library of processes and product components is made available to the user based on the skills of the system and pre-existing system knowledge. When adding the step, the logic engine collates those states necessary to execute these steps by inferring the relevant *ResourceSkills* and inserts them into the control code. Finally, the removal of a process step involves the opposite process as compared to insertion where those *ResourceSkills* associated with the process step are deleted to generate the final control code.

Once the user has completed the manipulation of the high level process plan, the OWL file is converted back into an XML file that is compliant with the engineering tools using GATE. Throughout this process, the control code was invisible to the process planner. This reduces the complexity of making changes and as a consequence, the errors associated with making them.



```
Algorithm SwapProcess
  Input: Process XML, Ontology RDF, Original StepID O, Target StepID T ,
New Process N, Delete StepID D.
  Output: Process XML with new process sequence.
  If XML.size = 0 return null
  RDF ← XML //Generate ontology from XML
  For each item in RDF, do
    If item [ProcessID] = O, then
      Origin ← item //Get original process sequence
    If item [ProcessID] = T, then
      Target ← item //Get target process sequence
  Temp ← Target, Target ← Origin, Origin ← Temp //swap process
  item = NewItem[N] //formalise process based on RDF rules
  RDF [NewID] ← item //insert N to RDF
  deleteItem  ← RDF [D] //navigate delete processes
  RDF ← RemoveItem[deleteItem] // remove item from RDF
  XML ← RDF //Generate new XML from RDF ontology
  Return XML
```

Figure 3-20 Algorithm for executing changes

3.9 *Chapter Summary*

This chapter opened with an overview of the methodology describing the general approach for creating domain ontologies, the need for a Skill model, and brief overview of vueOne. Following this, a detailed description of the respective domain ontologies as well as the Skill model was presented which form the foundation of this work. Next, the enrichment of the ontological models with rules was discussed. The early part of this chapter set the groundwork as to what the respective domain models looked like and how they were linked so that it would be clear to the reader how the exploitation of these ontological models would be carried out and how they would fit within a broader workflow.

After establishing this, the chapter described the inconsistency management aspect in three parts. The author proposed that the Resource domain is the most complex of the three domains and also has the greatest value associated with it. Therefore, the other domains as well as the Skill model need to understand its status and capabilities so that when new products are introduced and new process plans are generated, these can be checked with respect to Resource domain capability. This argument formed the first part of the inconsistency management section while also describing general queries that could be implemented to help support the capability checking. The second part described a method for identifying sequence inconsistencies using the Skill model. The third part established a framework that brought all of the elements of the broader methodology together with a view to allow sequence changes to be made (on the basis that an inconsistency has been identified) by manipulating APP information within the Process domain. This elaborated on the power and the need of the Skill model which has not been used in such a way in the literature. To achieve this, a key contribution was made which extended the data model of a set of component-based virtual engineering tools (vueOne – see Table 3-3) with the necessary concepts identified through systematically examining previous work within a similar context. The following chapter tests the queries and framework presented in this chapter on some use cases to validate the approach.

4 Application evaluation through Case Studies

4.1 *Introduction*

In the methodology chapter, inconsistency management through the use of the PPR ontology and Skill model was described. In this chapter, the ideas and approaches are tested through case studies. First the “capability checking” aspect of inconsistency management is checked via a case study that uses an engine assembly station. As the scope of the thesis and the methodology is then focused on supporting assembly process planning activities and their link to machine control software, the second case study focuses on testing this aspect. A fuel cell assembly is described using a high level and a low level description which is compared to machine control code to demonstrate how, regardless of original description language or granularity, a connection can be made between the respective descriptions/models (see Figure 3-19). Then two new fuel cell product variants are introduced and the logic of the assembly machine is modified accordingly.

4.2 *Case 1 - Checking manufacturing resource capability with respect to product and process requirements*

4.2.1 System description – engine assembly station

The case study for this part of the work is an assembly station from an engine assembly line of a large UK based engine manufacturer. Figure 4-1a describes the process that the station executes, the objective of which is to carry out a process known as a “nut running operation”. The outcome is to affix the engine oil pan to the main engine block. The process is summarised as follows. The engine arrives at the station on a conveyor. A data tag is read at which point the engine is clamped, and lifted to the nut runner. Then, the nut runner actuates, tightening all bolts simultaneously. Once completed, the engine is lowered and rotated, and then lowered again onto the pallet. Finally, the engine is unclamped and transported to the exit on a conveyor.

Figure 4-1b is a screenshot of the vueOne toolset’s core component editor module. The set of components is described on the left, and an example of a component’s state transition diagram (in this case, a clamp) is adjacent to it. The 3D model is to the right of the figure, while below it the cycle timing diagram is present, which is automatically generated from the data in the state transition diagram. Information from the virtual model including the components and the sequence were instantiated into the ontological model manually for this case study. The concepts that exist within the ontology and not in the original implementation of the engineering tools e.g. Skill (see Table 3-3), were also added manually to the ontology with the knowledge that such information could exist within the engineering toolset in the future. The author had process documentation from the automotive manufacturer which was used to populate the Process domain. Full product information was not available, but there were sufficient details to describe the product at the stage of completion for the station modelled. To maintain commercial confidence agreements, some information from the virtual model has been hidden, however this does not undermine the proof-of-concept presented in this research.

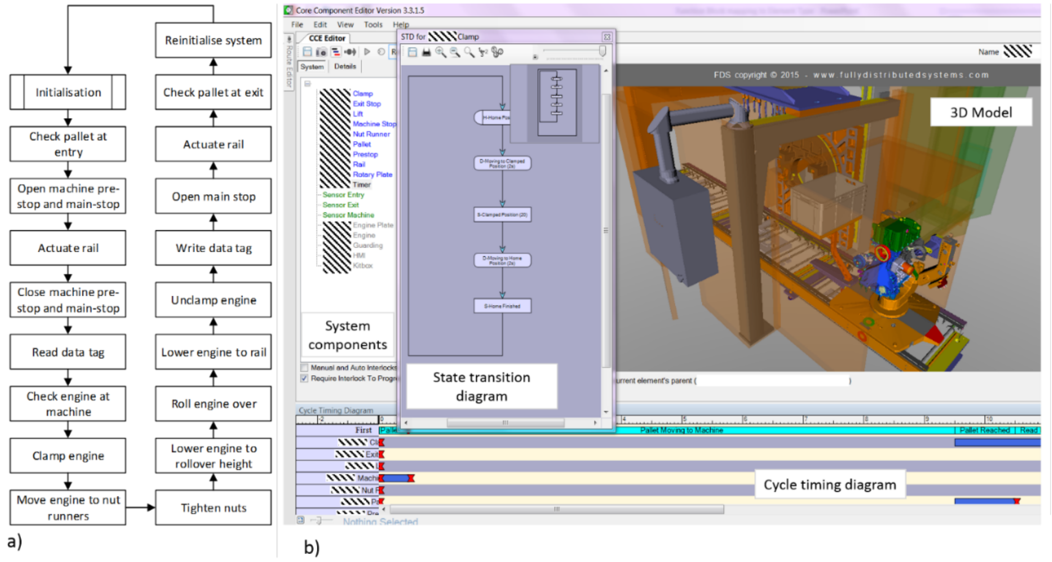


Figure 4-1a) Machine sequence, and b) annotated screenshot of assembly station within the vueOne engineering environment

4.2.2 Experimental Setup

Figure 4-2 illustrates how the ontology is extended to allow a check to be made for the system illustrated in Figure 4-1. The *BoltHeads* is a subclass of *ProductComponent* and this is instantiated with an instance of *BoltHeads* with details such as the number. Information regarding the *BoltHeads* is linked to the *EngineBlock* and *OilPan* (not illustrated in Figure 4-2) through the *hasLiaison* property and instantiates the class of *ScrewFitLiaison*. The *ScrewFitLiaison* class is linked to the Skill model via the Process Domain through a *NutRunning Process*. Note that the realisation of a liaison, as per the definition in the methodology, is exactly that instance of *Process* that describes its fulfilment. This allows the linking to a generic description of the action required which is *BoltTightening*. This same action is realised from the assembly station via the *NutRunner ControlComponent* through a specific state that exists within this component's STD. The

NutRunner actuator component has *NutRunnerHeads* fitted to it which form an instance of *NonControlComponent* and *hasEquivalentSkillContext* with the *ProductComponent* instance of *BoltHeads* (via the *BoltHeads* class). The full sequence associated with the process is represented within the *ProcessComponent* as the *NutRunningSequence*. The reader is reminded that this is the *ResourceView ProcessComponent* and the *ProcessView ProcessComponent* is not used in this part of the work.

4.2.3 Query 1 – Determining Resource capabilities with respect to Product requirements

The objective of Query 1, as discussed in the methodology chapter, is to determine whether a machine meets a product's requirements. The route that the general query would need to follow has already been illustrated and discussed in (ref Fig no 14 from methodology). In order to test whether the query would be able to deliver the results required, the contextual information available regarding product components and machine components was instantiated into the Skill model. In this example, due to the objective of the process being to bolt the oil pan to the engine block, the number associated with the number of bolts in the product is compared with the number of bolt heads in the machine. This is so that, if and when a new product variant was introduced with a different number of bolts, the capability of the machine could be checked with respect to the new requirement. This information could be enriched with process parameters such as the number of turns and the torque, due to the extensible nature of the model. The fully instantiated model is illustrated in Figure 4-3 as a screenshot of Protégé.

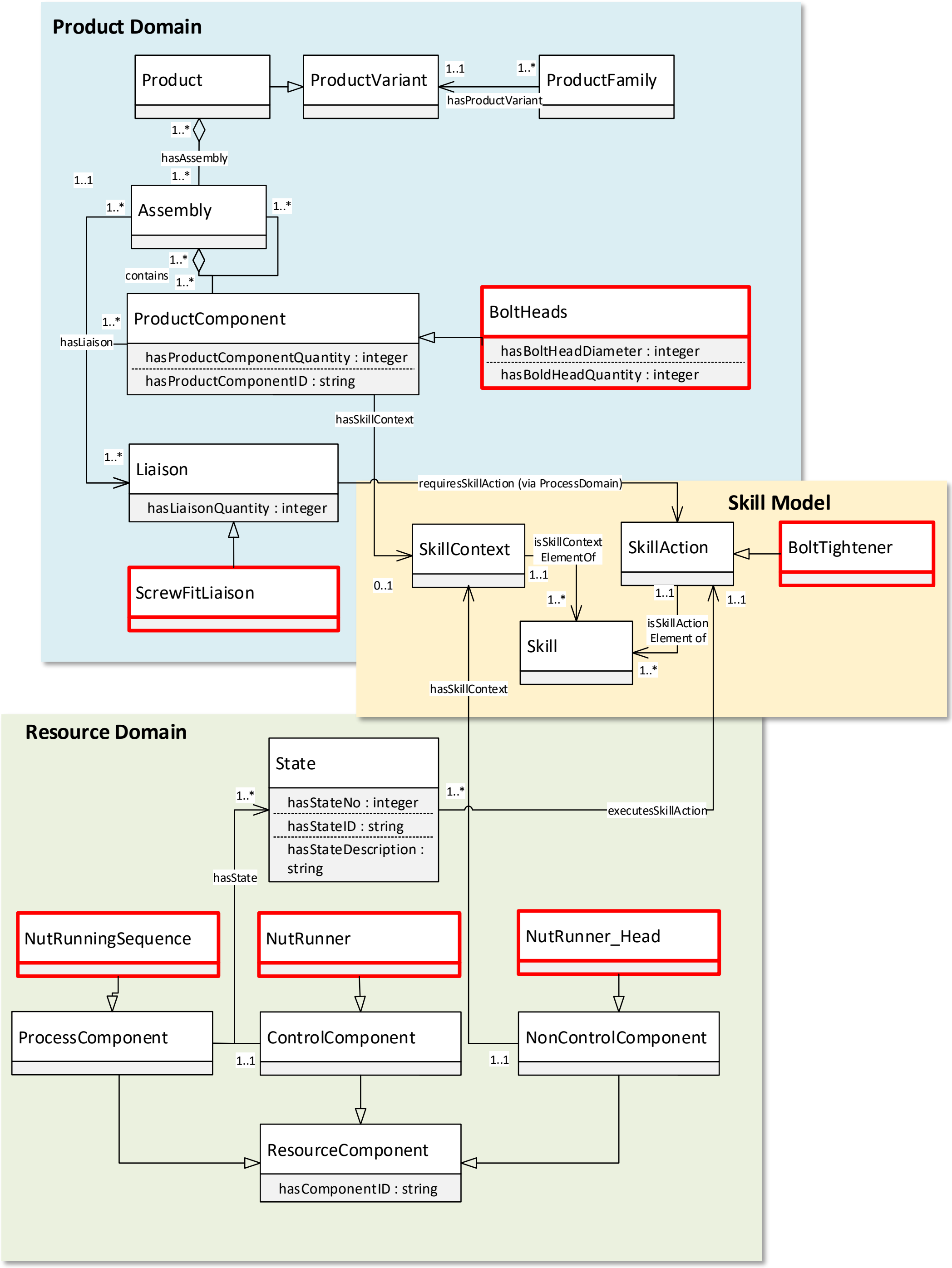
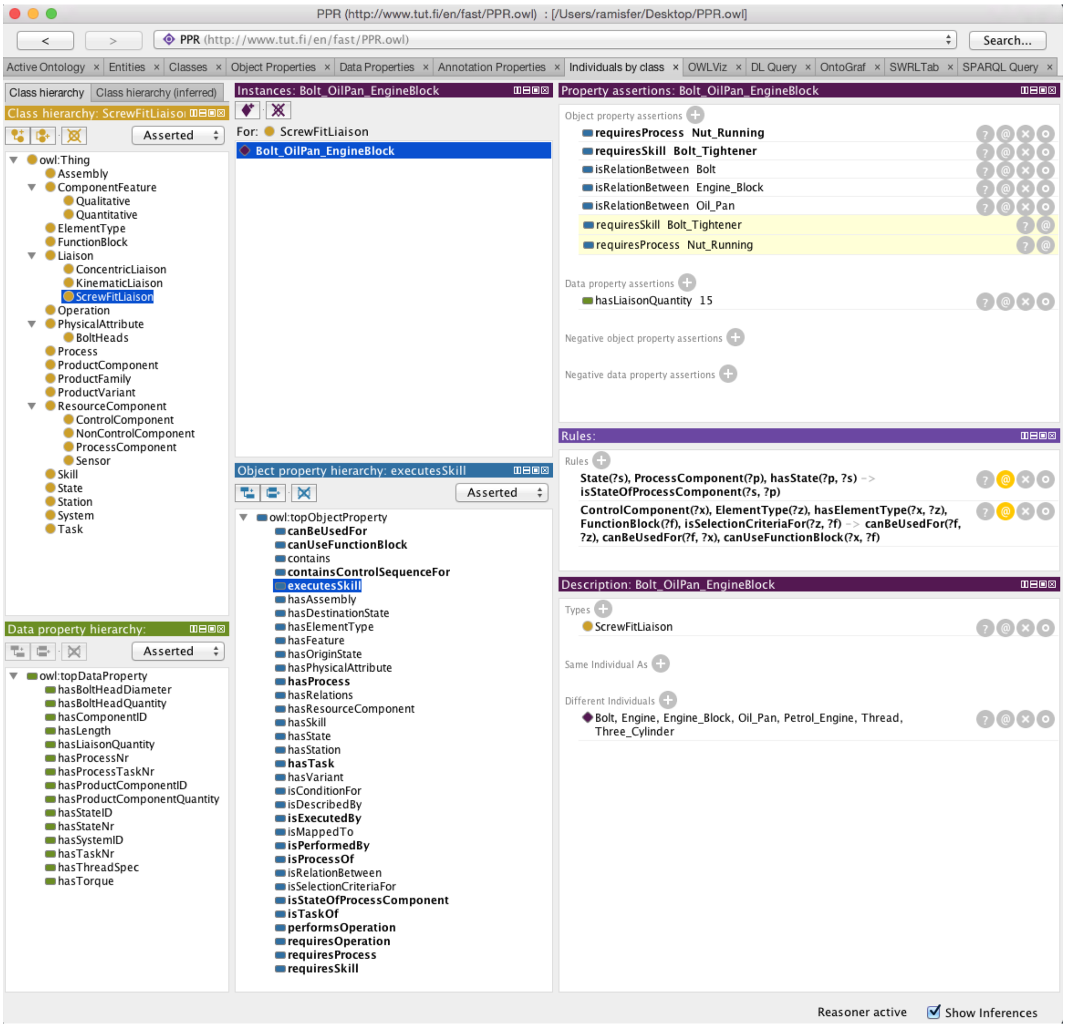


Figure 4-2 High level view of additional concepts added to model engine assembly station



Property assertions: Bolt_OilPan_EngineBlock

Object property assertions
- requiresProcess Nut_Running
- requiresSkill Bolt_Tightener
- isRelationBetween Bolt
- isRelationBetween Engine_Block
- isRelationBetween Oil_Pan
- requiresSkill Bolt_Tightener
- requiresProcess Nut_Running

Data property assertions
- hasLiaisonQuantity 15

Negative object property assertions

Negative data property assertions

Rules:

```
State(?s), ProcessComponent(?p), hasState(?p, ?s) -> isStateOfProcessComponent(?s, ?p)
ControlComponent(?x), ElementType(?z), hasElementType(?x, ?z), FunctionBlock(?f), isSelectionCriteriaFor(?z, ?f) -> canBeUsedFor(?f, ?z), canBeUsedFor(?f, ?x), canUseFunctionBlock(?x, ?f)
```

Description: Bolt_OilPan_EngineBlock

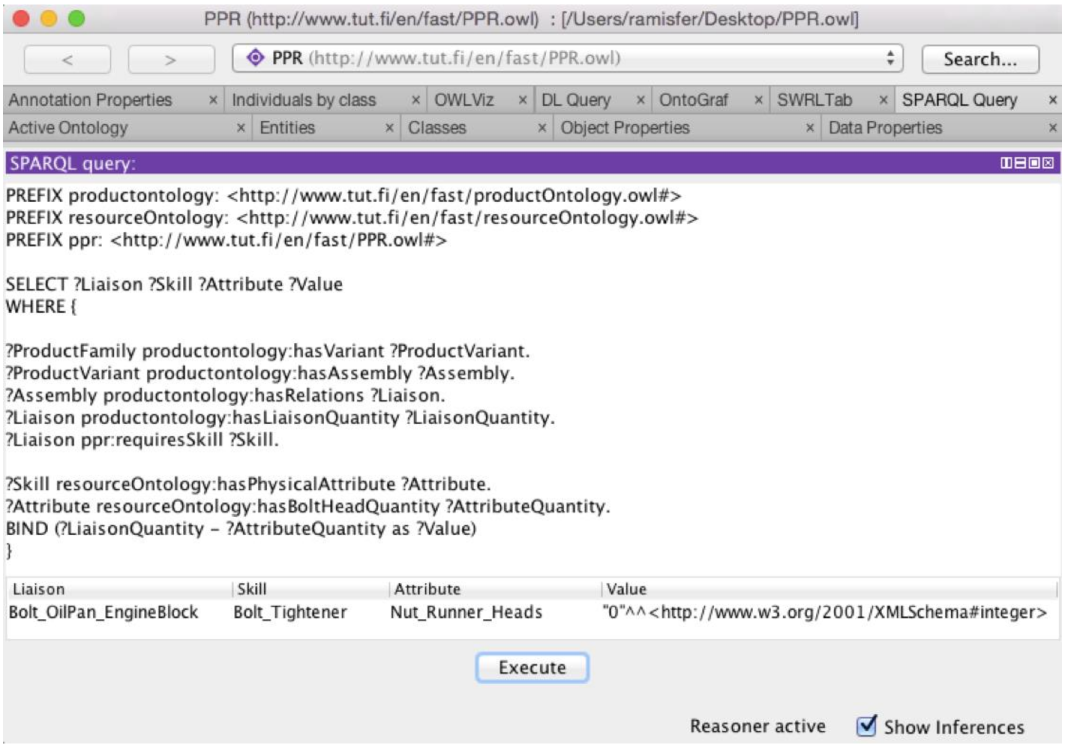
Types
- ScrewFitLiaison

Same Individual As

Different Individuals
- Bolt, Engine, Engine_Block, Oil_Pan, Petrol_Engine, Thread, Three_Cylinder

Figure 4-3 Protégé screenshot showing how the SWRL rules infer that the "Bolt-OilPan-EngineBlock" liaison infer the requirement of BoltTightener SkillAction and NutRunning Process (highlighted in yellow)

Although the generic description of Query 1 in the methodology chapter was focused on ascertaining Product-Resource capability consistency at a high level. This case study explores the expressive power of ontologies and data manipulation in more detail. As such, a query was created that not only checked whether the Resource was generally capable i.e. that the *BoltTightening* action existed, but also whether the contextual aspect of the skill was consistent i.e. does the Resource have a sufficient number of *NutRunner_Heads* to realise the number of *ScrewFitLiaisons*. As such, the query as illustrated in Figure 4-4 was created and the results are presented here also. Note that the structure of the query follows largely the same structure with respect to the routing as compared to the generic version of Query 1 given in chapter 3. However, the key difference is a simple mathematical calculation to determine the difference in the number of *NutRunner_Heads* and *ScrewFitLiaisons*. As the difference is "0" it is observed that no difference between capabilities exist and therefore the system is fully capable.



```
PREFIX productontology: <http://www.tut.fi/en/fast/productOntology.owl#>
PREFIX resourceOntology: <http://www.tut.fi/en/fast/resourceOntology.owl#>
PREFIX ppr: <http://www.tut.fi/en/fast/PPR.owl#>

SELECT ?Liaison ?Skill ?Attribute ?Value
WHERE {

?ProductFamily productontology:hasVariant ?ProductVariant.
?ProductVariant productontology:hasAssembly ?Assembly.
?Assembly productontology:hasRelations ?Liaison.
?Liaison productontology:hasLiaisonQuantity ?LiaisonQuantity.
?Liaison ppr:requiresSkill ?Skill.

?Skill resourceOntology:hasPhysicalAttribute ?Attribute.
?Attribute resourceOntology:hasBoltHeadQuantity ?AttributeQuantity.
BIND (?LiaisonQuantity - ?AttributeQuantity as ?Value)
}
```

Liaison	Skill	Attribute	Value
Bolt_OilPan_EngineBlock	Bolt_Tightener	Nut_Runner_Heads	"0"^^<http://www.w3.org/2001/XMLSchema#integer>

Reasoner active ☑ Show Inferences

Figure 4-4 SPARQL query and results for Query 1

The results from Query 1 show how it is possible to link product data features with the capabilities of resources to provide a mechanism for reconfiguration should requirements or capabilities change. However, there are several shortcomings of the proposed approach. One of the major issues is whether or not the user, who in this case would most likely be the product designer, has awareness of the knowledge available in the ontology. This would require that either, a considerable training exercise to communicate what can be queried is undertaken, or that the user's engineering tool is connected to the ontology. As a result, when the relevant inconsistency is identified, the user is notified from within the engineering environment. This would of course require further software development and the question would be raised as to whether the additional human resources required to maintain this connection would outweigh the benefits of seamless data model integration.

In addition, depending on the nature of the skill being assessed, the resultant processing will be different. For example, in the case of a load limit of a *ControlComponent* e.g. a robot, any value of mass of the product less than the limit would necessitate a result that would indicate to the user that the requirements were consistent with the capabilities. In another case, an assessment would need to be made based on a range. For example, a pneumatic gripper has an upper and lower bound for the size of component that it can handle. Provided a given *ProductComponent* was within the range, the user could be notified that capabilities/skills meet requirements. Therefore, a beneficial extension of the query would be to extend the ontology with additional rules that check the type of skill being assessed e.g. limit, range, difference etc. and then consider the result accordingly. In some sense, this was addressed in (Järvenpää, 2012) however the work was more focused on aggregating and matching capabilities rather than a detailed analysis of the numerical values associated with them.

Finally, it would be of benefit to the use if the numerical result generated by the query was pre-processed before printing. An example for the pre-processing that could be achieved for Query 1 is presented in Table 4-1. This post-processing and even the query itself does not need to exist within Protégé which is a relatively limiting environment. The JENA framework provides much more flexibility and due to its Java implementation would allow the more complex data processing to be carried out more readily.

Table 4-1 Pre-processing example for Query 1

Query result (hasLiaisonQuantity – hasBoltHeadQuantity)	Printed result	Interpretation
0	TRUE	the difference in the number of engine oil pan bolts and nut runner heads equals zero
Negative number	EXCESS SKILL	there are more nut runner heads on the machine than there are engine oil pan bolts. This could allow the designer to question design validity i.e. there may be insufficient bolts to hold the oil pan. On the other hand appropriate preparations could be made to modify the machine triggering an engineering change.
Positive number	EXCESS PRODUCT	there are fewer nut runner heads on the machine than there are engine oil pan bolts. Again, this allows the designer to question the design
Null	NULL PRODUCT	the required data does not exist in the product domain
Null	NULL RESOURCE	the required data does not exist in the resource domain
Null	FALSE	data does not exist in the product or resource domain

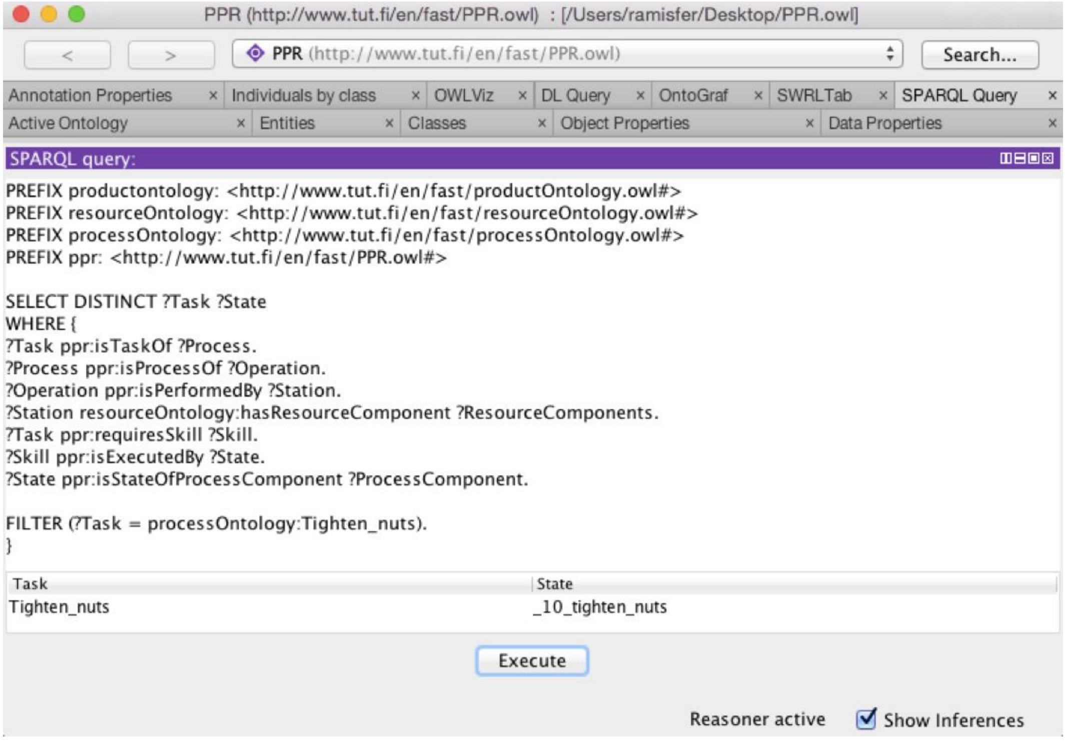
It is important to note that the "EXCESS SKILL" result that could be generated does not necessarily represent a design flaw, in fact it could simply be due to overcapacity within the system for flexibility reasons. However, the ability to know that there is a difference is simply an exchange of knowledge between the Resource domain and the Product domain.

With this knowledge in hand can allow the relevant stakeholders to make an informed decision where before there would have been a lack of transparency that a decision needed to be made at all.

4.2.4 Query 2 – Aligning Resource capabilities with Process Requirements

Query 2 examines the connectivity between the Process and Resource domains. When making a change to a process, it is often not clear how a piece of control logic relates to it, requiring controls experts, and thus increasing the length of the re-engineering process. This is due to the discrepancy between how different domains of an organisation work and operate. Although in the example presented for this case study, the naming convention between the process description in the Process Domain and the machine logic in the Resource Domain has been kept consistent it does not follow that this is also true within an industrial environment.

Query 2 is written such that it checks, for a specific instance of *Task*, whether there is a relevant state that executes it within the *ResourceView ProcessComponent*. The query and result is presented in Figure 4-5. This result demonstrates that it is possible, through the use of ontological models coupled with virtual engineering tools, to check Product domain and Resource domain compatibility. Note that the information concerning the states of the machine was derived, albeit manually, directly from the engineering tool model. In the implementation of this work, the user would manually need to work through each instance of *Task*. Thus, one of the shortcomings of this approach is that if there is an instance of *ProcessComponent* that has a *State* that executes a *Skill* and a given *Task* requires this *Skill* in more than one step, then the resulting inference may be incorrect. For example, consider a product that requires a bolt to be tightened in the early stage of an operation, and then again at the final stages. While the system may have bolt tightening capabilities, due to mechanical constraints, the station may be unable to fulfil both bolt tightening requirements. However, the query as it stands would infer that the station would be capable. This highlights the reason why the contextual information is important. Thus, it is necessary to extend the query such that it produces a true result that is consistent with the real world.



```
PREFIX productontology: <http://www.tut.fi/en/fast/productOntology.owl#>
PREFIX resourceOntology: <http://www.tut.fi/en/fast/resourceOntology.owl#>
PREFIX processOntology: <http://www.tut.fi/en/fast/processOntology.owl#>
PREFIX ppr: <http://www.tut.fi/en/fast/PPR.owl#>

SELECT DISTINCT ?Task ?State
WHERE {
?Task ppr:isTaskOf ?Process.
?Process ppr:isProcessOf ?Operation.
?Operation ppr:isPerformedBy ?Station.
?Station resourceOntology:hasResourceComponent ?ResourceComponents.
?Task ppr:requiresSkill ?Skill.
?Skill ppr:isExecutedBy ?State.
?State ppr:isStateOfProcessComponent ?ProcessComponent.

FILTER (?Task = processOntology:Tighten_nuts).
}
```

Task	State
Tighten_nuts	_10_tighten_nuts

Figure 4-5 SPARQL query and result for Query 2

4.2.5 Summary of Case 1

The results for Case 1 have demonstrated that:

i) the comparison of product requirements with resource capabilities can be achieved by extending the ontology with the relevant classes and making small modifications to the generic query

ii) a knowledge base can be used to integrate process planning with machine logic at the state level to ensure consistency.

Furthermore, the work has illustrated how such an approach complements existing industrial practices by presenting a methodology for how the queries presented in this research would exploit the knowledge-base to support design and engineering teams across the product realisation domains.

4.3 *Case 2 – Connecting assembly process plans at different granularities to machine control software*

4.3.1 System description – fuel cell assembly station

The hydrogen fuel cell is an electrochemical device that converts hydrogen and oxygen into electricity and water. There continues to be real-world implementations of this technology, particularly in the automotive sector by the likes of Toyota, Nissan, and more. Hydrogen fuel cells are a promising technology to facilitate in the decarbonisation of energy across industries ranging from portable power through to stationary and back-up. Despite their inherent power flexibility (attributed to modularity) nuanced design changes emerge to satisfy the specific needs of the respective markets. One of the consequences of these design changes is inevitably the change in assembly sequence. In addition, during the research and development phase of fuel cells, it is necessary to experiment with different sequences to ascertain the impact on performance.

During the course of this PhD research project, the author has had the opportunity to be involved with a number of industrial projects. Two of which have been focused on the manufacturing and assembly of hydrogen fuel cells. As a result, the author has built an appreciation of the nuances of fuel cell assembly, particularly within the context of specific designs. However, due to confidentiality clauses and the sensitive nature of Intellectual Property (IP) associated with the output of projects, the application of the approach described in this research is limited to abstracted version of products, processes, and systems. Despite this abstraction process, the author still claims that the approach described in Chapter 3 can be validated and thus demonstrates a contribution to the body of knowledge as to how process plans described at differing granularities can be mapped to machine logic.

The general structure of a fuel cell stack is presented in Figure 4-6a, and a single cell with unique components IDs and component liaison IDs in Figure 4-6b, respectively. The focus area for the checking domain model consistency between the assembly process plan (APP) and machine control software (MCS) is outlined in red in Figure 4-6c. The sub-assembly highlighted is referred to as a half cell and is a mirror image of the relationships between components C_5, C_6, and C_7. As a whole, Figure 4-6 summarises the data required to instantiate the Product Domain for this part of the work.

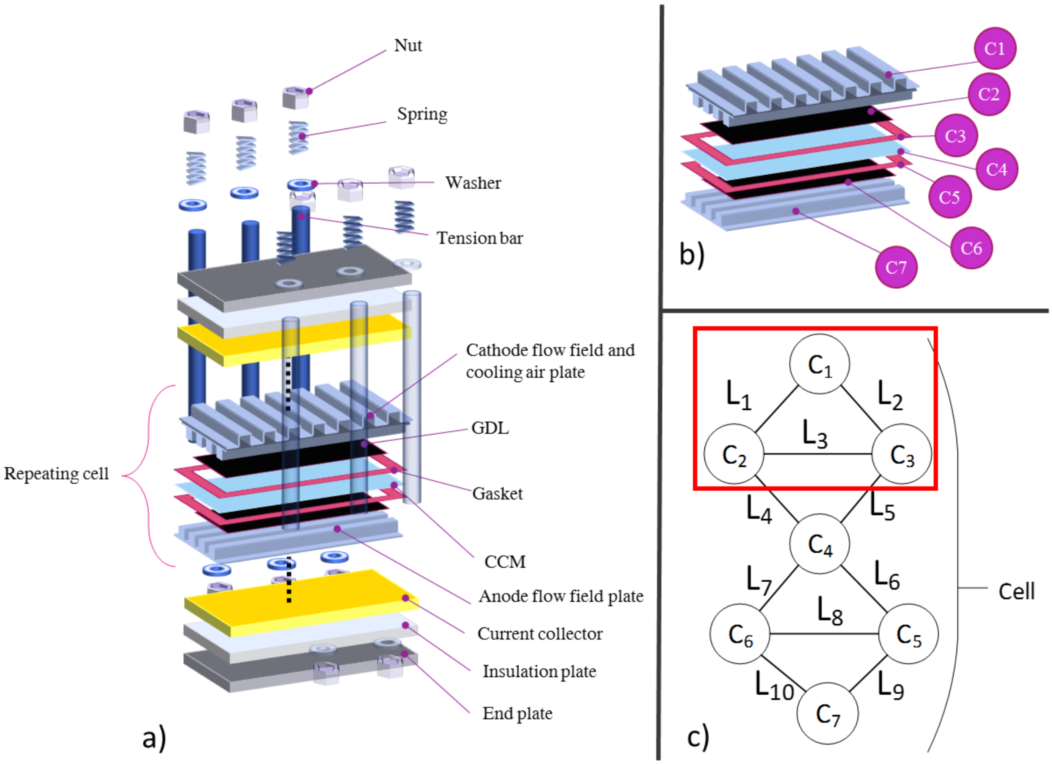


Figure 4-6a) Illustration of fuel cell stack, b) fuel cell component IDs, and c) undirected graph with focus area for case study in red

The process sequence for realising the half-cell assembly illustrated in Figure 4-6c is described in two levels of granularity in Figure 4-7a as “Description 1” and “Description 2”. Each step in “Description 1” aligns with the definition of the *Process* class, on the other hand “Description 2” describes a more decomposed set of processes and is thus more aligned to the *Task* class description. The author acknowledges that the process could also be described in other ways to realise the same relations of the product at a common level of granularity to the descriptions presented in Figure 4-7a. In order to keep the case study clear and concise, only those descriptions illustrated have been tested within this case study.

The assembly system used to assemble the half-cell assembly is illustrated in Figure 4-7b together with MCS in SFC format. This describes only the behaviour of the sequence logic of the system and is thus equivalent to the *ResourceView ProcessComponent*. This diagram has been recreated from the STD that is generated from the vueOne engineering tools to improve readability. Note that the additional logic of the respective actuators has not been included in this diagram but do exist within the engineering model and a physical system.

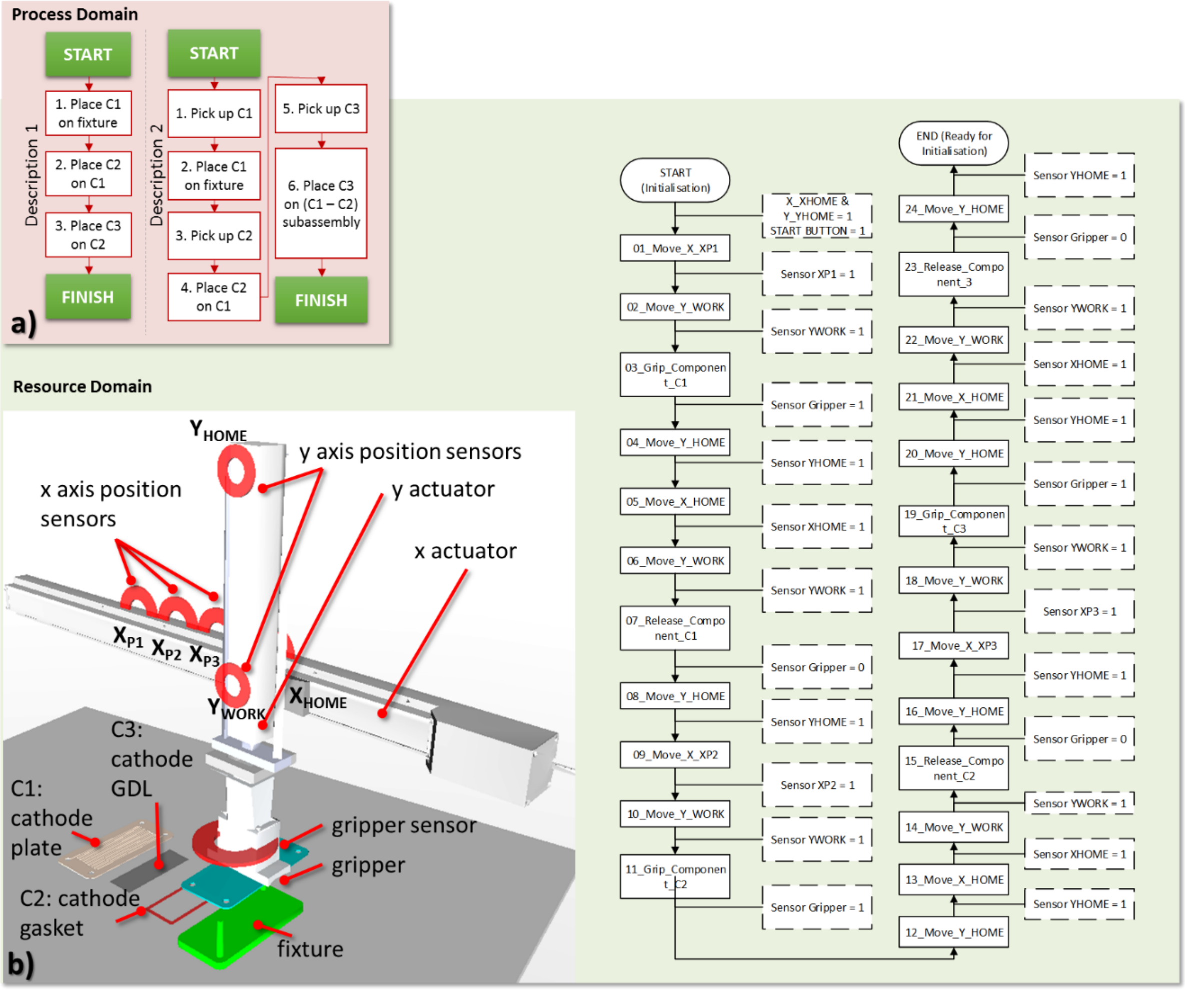


Figure 4-7a) Two levels of process description granularity for half-cell assembly, and b) the assembly system used represented in the vueOne engineering tools together with the ResourceView ProcessComponent

The workflow in the case of this section of the work is that the process planner would receive the product information as illustrated in Figure 4-6 within the broader workflow that is presented in the methodology chapter. On receiving this information and through discussion with the product designer an assembly sequence would be derived. This information would then be passed onto the mechanical engineer who would, through support from the aforementioned domain stakeholders, design a machine. The mechanical engineer would be supported by the controls engineer who would derive the control code for the machine, and a combination of efforts from stakeholders in the Resource Domain would result in its physical instantiation. It is proposed that to support the activities of the Resource Domain the vueOne engineering tools would be used and thus the virtual model as illustrated in Figure 4-7b would be created. As opposed to the controls engineer creating control code from scratch, it could be automatically generated from the virtual model as has already been described and proven in (Ahmad, 2014). This automatic code would lend itself to the more integrated engineering approach being described in this thesis as it would inherently conform to a structure and not the style of a given software/controls engineer. Despite this difference, the information content describing the control sequence would be expected to be the same regardless of whether the code is manually or automatically created. Therefore, it is proposed that the APP to MCS approach would be applicable to either scenario. Regardless, the problem persists in that the complexity of the control level description (*ResourceView ProcessComponent)* is difficult to validate with the high level description (*ProcessView ProcessComponent)*. Thus, the objective of this section of the case study are twofold:

i) how APP descriptions at different abstraction levels can be checked for consistency with respect to **capabilities** described in MCS

ii) how APP **sequence** requirements can be checked for consistency with behaviour described in MCS

Note that objective 1 in this case is also resolved in the first case study, however the validation of the methodology is enhanced through testing and demonstration on an entirely different case application. Furthermore, in the first case study the process description was already consistent with the resource description logic and so the test was only to ascertain that skills were compatible. In this case the difference is a change in the granularity of descriptions and ascertaining whether a consistency check can still be made.

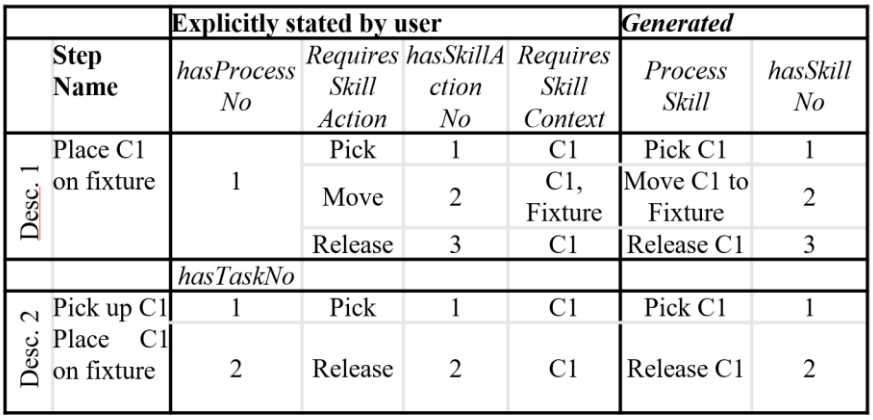
4.3.2 Transforming domain descriptions to ontological models

Table 4-2 and Table 4-3 are extracts of the explicit relationships declared by the user made within their respective engineering environments for the Process Domain and the Resource Domain respectively. More specifically, it is possible to declare these explicit relationships within the vueOne engineering tools due to the increased descriptive power of the tools. However the implementation of the extended data model has not been realised.

Note that these extracts are all functionally equivalent in that they are achieving the objective of moving component C1 from its initial position and placing it on the fixture. Furthermore, it is important to highlight that step 1 in “Description 1” is functionally equivalent to the sum of step 1 and step 2 in “Description 2”. However, despite the latter being a more decomposed version of the former, the resulting number of *ProcessSkill* instances is fewer. This is to demonstrate to the reader that this work cannot address the problem of maintaining equivalent descriptions in an absolute sense despite both descriptions using the same decomposition approach. Rather, it demonstrates how the meaning of the descriptions in both source models can be compared and checked for inconsistencies, regardless of how subtle differences between how the user may wish to describe the process may emerge.

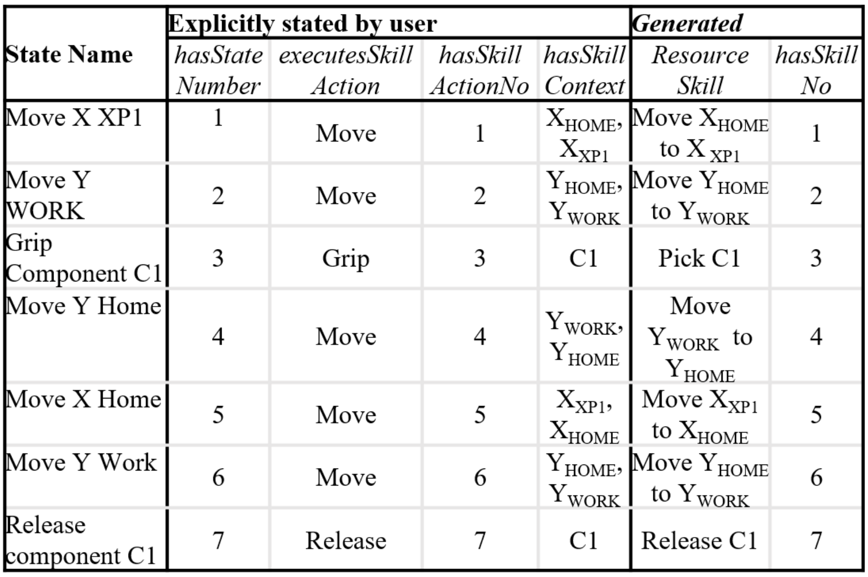
In the process of declaring instances the user is able to decompose their description in a systematic way that aligns with the standard terminologies for Skill while retaining the semantics associated with contexts. Once the explicit aspects have been declared the instance associated with *requiresSkillAction* class and *requiresSkillContext* are aggregated to form *ProcessSkill* or *ResourceSkill* depending on the source domain. This aggregation process transforms the manually decomposed descriptions into human and machine readable descriptions that can be cross checked for consistency.

Table 4-2 Extract of explicit and generated mappings – Process Domain



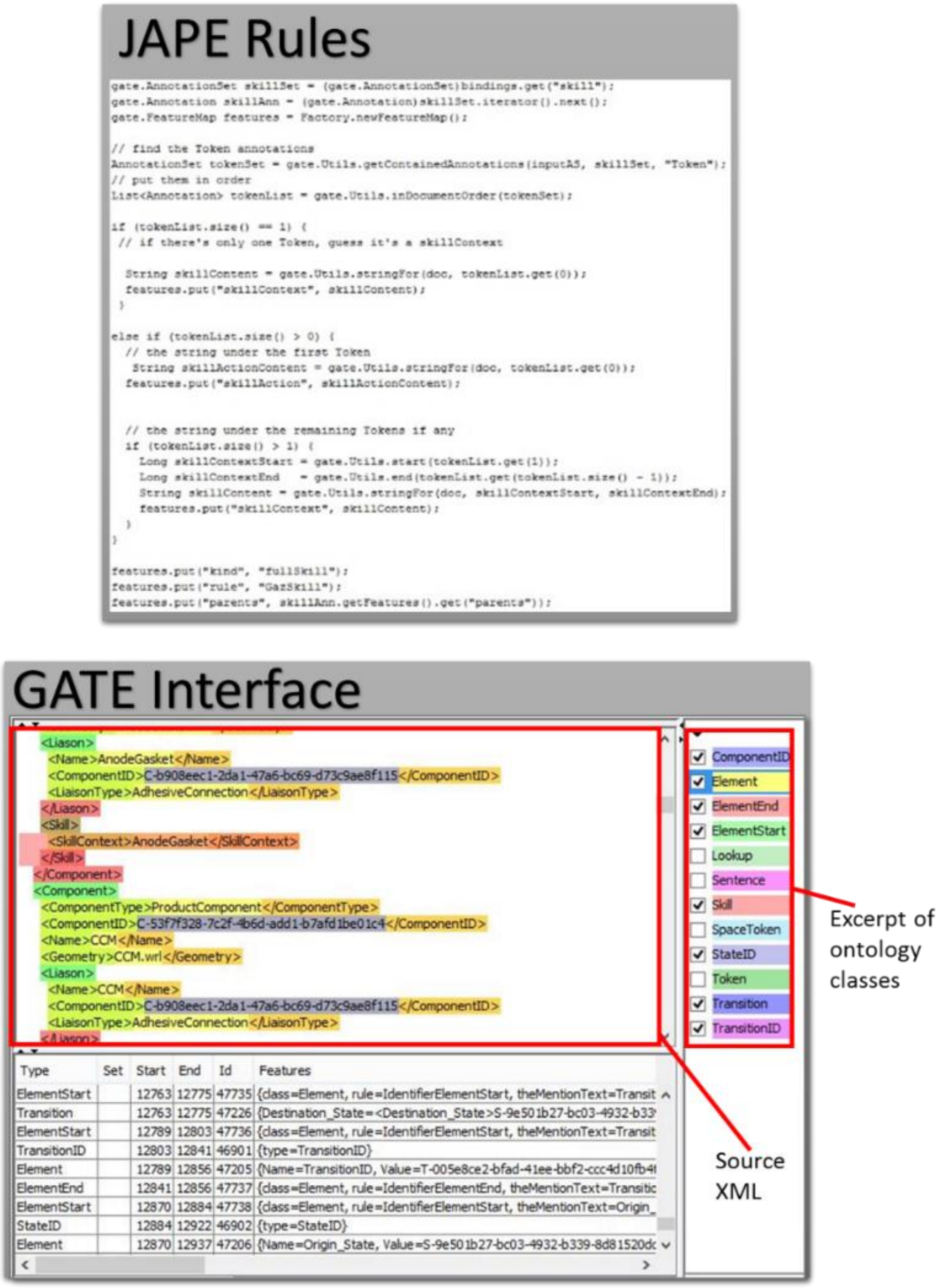
		Explicitly stated by user				***Generated***	
	Step Name	*hasProcess No*	*Requires Skill Action*	*hasSkillA ction No*	*Requires Skill Context*	*Process Skill*	*hasSkill No*
Desc. 1	Place C1 on fixture	1	Pick	1	C1	Pick C1	1
			Move	2	C1, Fixture	Move C1 to Fixture	2
			Release	3	C1	Release C1	3
		hasTaskNo					
Desc. 2	Pick up C1	1	Pick	1	C1	Pick C1	1
	Place C1 on fixture	2	Release	2	C1	Release C1	2

Table 4-3 Extract of explicit and generated mappings – Resource Domain



State Name	**Explicitly stated by user**				***Generated***	
	hasState Number	*executesSkill Action*	*hasSkill ActionNo*	*hasSkill Context*	*Resource Skill*	*hasSkill No*
Move X XP1	1	Move	1	X_{HOME}, X_{XP1}	Move X_{HOME} to X_{XP1}	1
Move Y WORK	2	Move	2	Y_{HOME}, Y_{WORK}	Move Y_{HOME} to Y_{WORK}	2
Grip Component C1	3	Grip	3	C1	Pick C1	3
Move Y Home	4	Move	4	Y_{WORK}, Y_{HOME}	Move Y_{WORK} to Y_{HOME}	4
Move X Home	5	Move	5	X_{XP1}, X_{HOME}	Move X_{XP1} to X_{HOME}	5
Move Y Work	6	Move	6	Y_{HOME}, Y_{WORK}	Move Y_{HOME} to Y_{WORK}	6
Release component C1	7	Release	7	C1	Release C1	7

The information that already exists in the vueOne engineering model was auto-instantiated into the ontology using JAPE rules which connected the XML file output to the relevant classes of the OWL model. GATE was used as the interface between the source XML (based on the XML tags) and the OWL model due to its ability to import OWL models. A screenshot of the JAPE rules and GATE interface is presented in Figure 4-8.



JAPE Rules

```
gate.AnnotationSet skillSet = (gate.AnnotationSet)bindings.get("skill");
gate.Annotation skillAnn = (gate.Annotation)skillSet.iterator().next();
gate.FeatureMap features = Factory.newFeatureMap();

// find the Token annotations
AnnotationSet tokenSet = gate.Utils.getContainedAnnotations(inputAS, skillSet, "Token");
// put them in order
List<Annotation> tokenList = gate.Utils.inDocumentOrder(tokenSet);

if (tokenList.size() == 1) {
 // if there's only one Token, guess it's a skillContext

  String skillContent = gate.Utils.stringFor(doc, tokenList.get(0));
  features.put("skillContext", skillContent);
 }

else if (tokenList.size() > 0) {
  // the string under the first Token
   String skillActionContent = gate.Utils.stringFor(doc, tokenList.get(0));
  features.put("skillAction", skillActionContent);


  // the string under the remaining Tokens if any
  if (tokenList.size() > 1) {
    Long skillContextStart = gate.Utils.start(tokenList.get(1));
    Long skillContextEnd   = gate.Utils.end(tokenList.get(tokenList.size() - 1));
    String skillContent = gate.Utils.stringFor(doc, skillContextStart, skillContextEnd);
    features.put("skillContext", skillContent);
  }
}

features.put("kind", "fullSkill");
features.put("rule", "GazSkill");
features.put("parents", skillAnn.getFeatures().get("parents"));
```

GATE Interface

```
    <Liason>
      <Name>AnodeGasket</Name>
      <ComponentID>C-b908eec1-2da1-47a6-bc69-d73c9ae8f115</ComponentID>
      <LiaisonType>AdhesiveConnection</LiaisonType>
    </Liason>
    <Skill>
      <SkillContext>AnodeGasket</SkillContext>
    </Skill>
  </Component>
  <Component>
    <ComponentType>ProductComponent</ComponentType>
    <ComponentID>C-53f7f328-7c2f-4b6d-add1-b7afd1be01c4</ComponentID>
    <Name>CCM</Name>
    <Geometry>CCM.wrl</Geometry>
    <Liason>
      <Name>CCM</Name>
      <ComponentID>C-b908eec1-2da1-47a6-bc69-d73c9ae8f115</ComponentID>
      <LiaisonType>AdhesiveConnection</LiaisonType>
    </Liason>
```

Type	Set	Start	End	Id	Features
ElementStart		12763	12775	47735	{class=Element, rule=IdentifierElementStart, theMentionText=Transit
Transition		12763	12775	47226	{Destination_State=<Destination_State>S-9e501b27-bc03-4932-b33
ElementStart		12789	12803	47736	{class=Element, rule=IdentifierElementStart, theMentionText=Transit
TransitionID		12803	12841	46901	{type=TransitionID}
Element		12789	12856	47205	{Name=TransitionID, Value=T-005e8ce2-bfad-41ee-bbf2-ccc4d10fb4
ElementEnd		12841	12856	47737	{class=Element, rule=IdentifierElementEnd, theMentionText=Transitic
ElementStart		12870	12884	47738	{class=Element, rule=IdentifierElementStart, theMentionText=Origin_
StateID		12884	12922	46902	{type=StateID}
Element		12870	12937	47206	{Name=Origin_State, Value=S-9e501b27-bc03-4932-b339-8d81520dc

☑ ComponentID
☑ Element
☑ ElementEnd
☑ ElementStart
☐ Lookup
☐ Sentence
☑ Skill
☐ SpaceToken
☑ StateID
☐ Token
☑ Transition
☑ TransitionID

Excerpt of ontology classes

Source XML

Figure 4-8 JAPE rules and GATE interface

To help illustrate how skills are assigned within the ontology and how the inter-domain connections are formed, Figure 4-9 represents the case information within the framework illustrated in Figure 3-19. The PPR ontology is instantiated with the data that has been presented in Table 4-2 and Table 4-3 based on the case illustrated in Figure 4-7. A manual process from the respective members of the team layer would explicitly define what constitutes a *SkillAction* or *SkillContext* within the Process domain description (*ProcessView*) and the control model in the Resource domain description (*ResourceView*). Nouns that exist within the Product domain such as *ProductComponent* or the Resource domain such as *Fixture* are automatically transformed into unique instances of *SkillContext* using the unique ID that is generated from the engineering tool and assigned a *hasSkillContext* relation with the respective instance. Due to this link, users can identify to what the *SkillContext* is being referred to as typically the unique ID could not be interpreted in isolation to reveal its source.

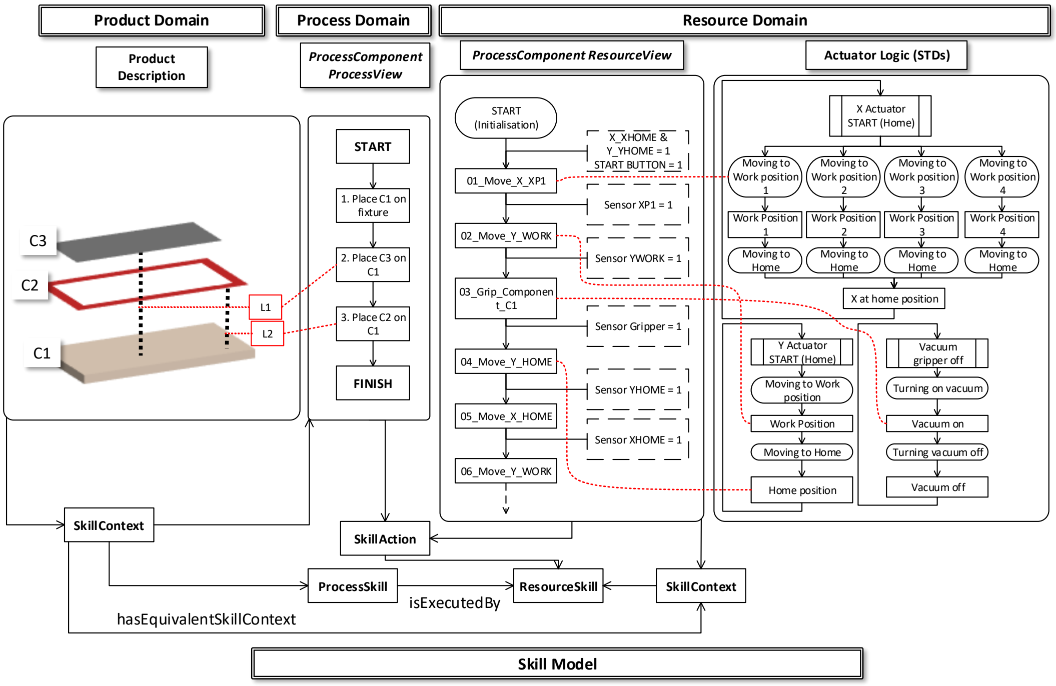
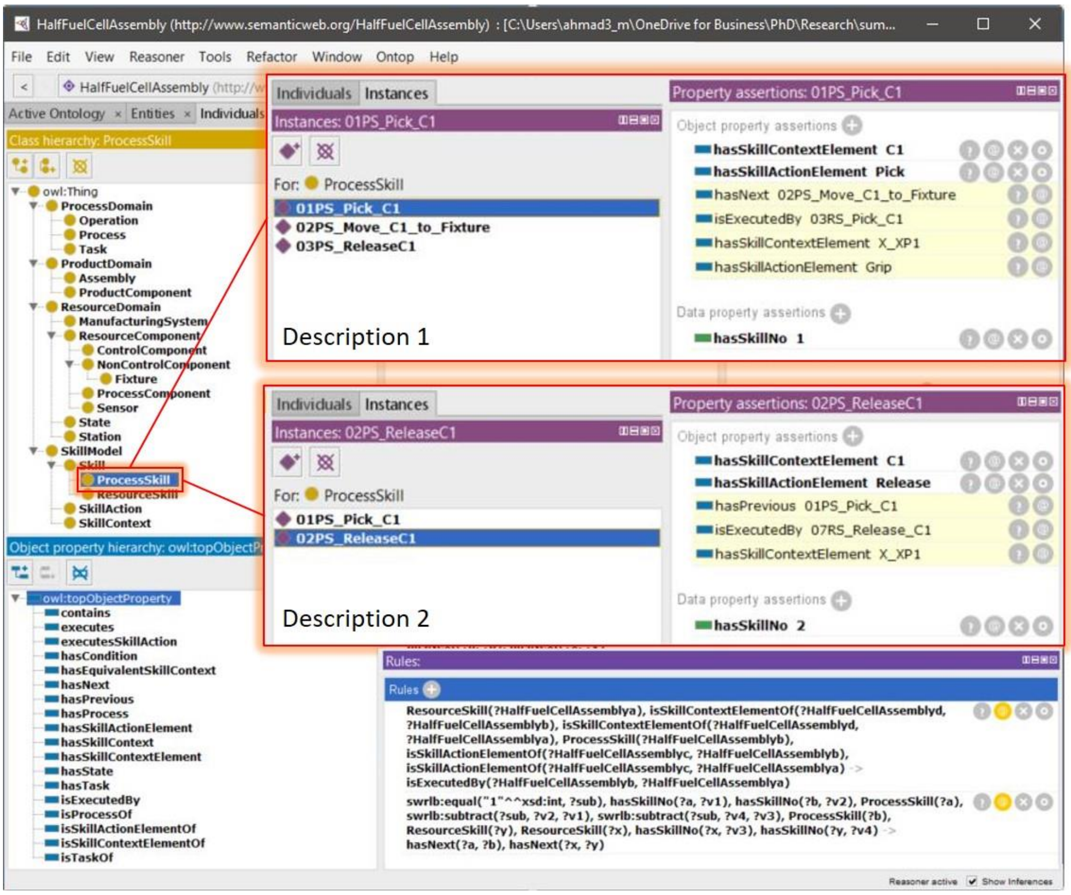


Figure 4-9 Assigning and mapping skills

4.3.3 Implementation and results

4.3.3.1 Objective 1

The Protégé screenshot presented in Figure 4-10 illustrates the implementation of “Description 1” and “Description 2” described in Table 4-2. New knowledge inferred from Rule 1 (Table 3-5) is highlighted in yellow. *Objective 1* of this part of the case study is therefore achieved as the figure demonstrates that regardless of the initial abstraction level presented in Table 4-2, the appropriate *ResourceSkill* that can execute it is still mapped. This demonstrates that the model is able identify whether there is a state in the MCS that is able to execute a given process step.



Instances: 01PS_Pick_C1

For: ProcessSkill

- 01PS_Pick_C1
- 02PS_Move_C1_to_Fixture
- 03PS_ReleaseC1

Property assertions: 01PS_Pick_C1

Object property assertions

- hasSkillContextElement C1
- hasSkillActionElement Pick
- hasNext 02PS_Move_C1_to_Fixture
- isExecutedBy 03RS_Pick_C1
- hasSkillContextElement X_XP1
- hasSkillActionElement Grip

Data property assertions

- hasSkillNo 1

Instances: 02PS_ReleaseC1

For: ProcessSkill

- 01PS_Pick_C1
- 02PS_ReleaseC1

Property assertions: 02PS_ReleaseC1

Object property assertions

- hasSkillContextElement C1
- hasSkillActionElement Release
- hasPrevious 01PS_Pick_C1
- isExecutedBy 07RS_Release_C1
- hasSkillContextElement X_XP1

Data property assertions

- hasSkillNo 2

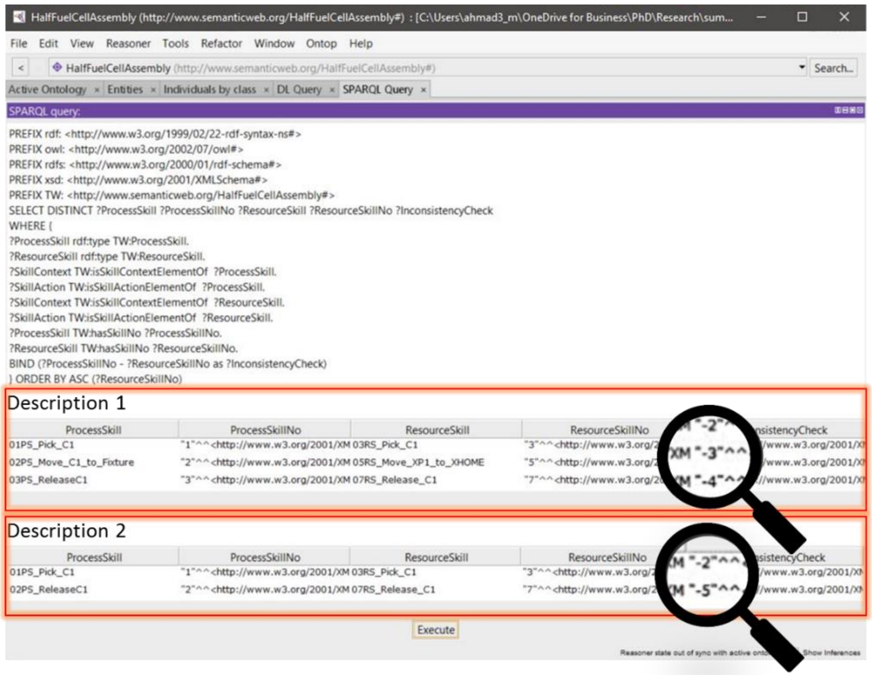
Rules:

```
ResourceSkill(?HalfFuelCellAssemblya), isSkillContextElementOf(?HalfFuelCellAssemblyd,
?HalfFuelCellAssemblyb), isSkillContextElementOf(?HalfFuelCellAssemblyd,
?HalfFuelCellAssemblya), ProcessSkill(?HalfFuelCellAssemblyb),
isSkillActionElementOf(?HalfFuelCellAssemblyc, ?HalfFuelCellAssemblyb),
isSkillActionElementOf(?HalfFuelCellAssemblyc, ?HalfFuelCellAssemblya) ->
isExecutedBy(?HalfFuelCellAssemblyb, ?HalfFuelCellAssemblya)
swrlb:equal("1"^^xsd:int, ?sub), hasSkillNo(?a, ?v1), hasSkillNo(?b, ?v2), ProcessSkill(?a),
swrlb:subtract(?sub, ?v2, ?v1), swrlb:subtract(?sub, ?v4, ?v3), ProcessSkill(?b),
ResourceSkill(?y), ResourceSkill(?x), hasSkillNo(?x, ?v3), hasSkillNo(?y, ?v4) ->
hasNext(?a, ?b), hasNext(?x, ?y)
```

Figure 4-10 Implementation of Description 1 and Description 2 in Protégé

4.3.3.2 Objective 2

Figure 4-11 illustrates the result of the inconsistency check for both “Description 1” and “Description 2”. The rationalisation for ascertaining whether an inconsistency exists (as discussed in the methodology chapter, see section 3.8.2) is demonstrated in the 5th column of the query results table in Figure 4-11. The difference between the *ProcessSkillNo* and the *ResourceSkillNo* is always negative indicating consistent descriptions. Furthermore, the figure also demonstrates that regardless of the different levels of process description abstraction levels, the consistency relative to the machine control logic description can be checked. This query fulfils *Objective 2* of this part of the case study as it demonstrates how the process planner’s model can be checked for consistency with the machine programme logic from the sequence perspective. However, only sequential processes have been checked for consistency. It is not uncommon for branched or parallel processes to exist in real machines. It is possible that the same methodology could be applied to resolve this issue also, with the caveat that those processes that exist within the respective branches having a standardised numerical coding associating with the states. At the initial design phase of the code, this may be possible. However, as the control code evolves, it may not be possible to maintain this standard because of its growing complexity and the lack of the software/controls engineer’s knowledge about the full code. This creates a stronger case for auto-code generation as it ensures that the control code follows a certain standard at all times.



HalfFuelCellAssembly (http://www.semanticweb.org/HalfFuelCellAssembly#) : [C:\Users\ahmad3_m\OneDrive for Business\PhD\Research\sum...
File Edit View Reasoner Tools Refactor Window Ontop Help
HalfFuelCellAssembly (http://www.semanticweb.org/HalfFuelCellAssembly#)
Search...
Active Ontology × Entities × Individuals by class × DL Query × SPARQL Query ×
SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX TW: <http://www.semanticweb.org/HalfFuelCellAssembly#>
SELECT DISTINCT ?ProcessSkill ?ProcessSkillNo ?ResourceSkill ?ResourceSkillNo ?InconsistencyCheck
WHERE {
?ProcessSkill rdf:type TW:ProcessSkill.
?ResourceSkill rdf:type TW:ResourceSkill.
?SkillContext TW:isSkillContextElementOf  ?ProcessSkill.
?SkillAction TW:isSkillActionElementOf  ?ProcessSkill.
?SkillContext TW:isSkillContextElementOf  ?ResourceSkill.
?SkillAction TW:isSkillActionElementOf  ?ResourceSkill.
?ProcessSkill TW:hasSkillNo ?ProcessSkillNo.
?ResourceSkill TW:hasSkillNo ?ResourceSkillNo.
BIND (?ProcessSkillNo - ?ResourceSkillNo as ?InconsistencyCheck)
} ORDER BY ASC (?ResourceSkillNo)
```

Description 1

ProcessSkill	ProcessSkillNo	ResourceSkill	ResourceSkillNo	nsistencyCheck
01PS_Pick_C1	"1"^^<http://www.w3.org/2001/XM	03RS_Pick_C1	"3"^^<http://www.w3.org/2	/www.w3.org/2001/X
02PS_Move_C1_to_Fixture	"2"^^<http://www.w3.org/2001/XM	05RS_Move_XP1_to_XHOME	"5"^^<http://www.w3.org/2	/www.w3.org/2001/X
03PS_ReleaseC1	"3"^^<http://www.w3.org/2001/XM	07RS_Release_C1	"7"^^<http://www.w3.org/2	//www.w3.org/2001/X

Description 2

ProcessSkill	ProcessSkillNo	ResourceSkill	ResourceSkillNo	sistencyCheck
01PS_Pick_C1	"1"^^<http://www.w3.org/2001/XM	03RS_Pick_C1	"3"^^<http://www.w3.org/2	/www.w3.org/2001/XM
02PS_ReleaseC1	"2"^^<http://www.w3.org/2001/XM	07RS_Release_C1	"7"^^<http://www.w3.org/2	/www.w3.org/2001/XM

Figure 4-11 Query and query results for inconsistency check for objective 2

4.3.4 Summary of Case 2

The results for Case 2 have demonstrated that:

i) The interaction and the mapping between the PPR domain via the Skill model allow consistency checks to be made with respect to capability across APP descriptions (at different levels of granularity) with MCS

ii) Sequence inconsistency checks between APP (at different levels of granualrtiy) can be made with MCS.

Within the broader workflow, one of the challenges would be to get the respective domain stakeholders to decompose their respective process steps or machine states as proposed. This would add additional workload to the user and thus may prevent acceptance. In addition, the issue of developing a GUI that interacts with the user has not been addressed in this research work but remains an essential part of the chain. It would be necessary to link said GUI both with the ontology as well as the engineering tool being used. This could be achieved through a framework like Apache Jena which is used in Case study 3 to support in the rectification of inconsistencies. Despite the challenges, one of the key issues within the context of differing semantics for common entities has been addressed through the *hasEquivalentSkillContextWith* object property which is a novel insight and the author has been unable to find a similar approach in the literature. It could be possible that as the knowledge within a given ontology increases and evolves, tools such as natural language processing and machine learning could be implemented. This would allow both the extraction of the necessary information from the respective engineering models, but also the automatic mapping of the aforementioned object property through inference to create a powerful, reusable knowledge base.

4.4 *Case Study 3 – Resolving inconsistencies*

Case study 3 is an extension of Case 2 in that it uses largely the same product, process, and resource information. The addition in this case is the introduction of product variants and a demonstration of how the respective semantic and semantic web technologies are used to complete the workflow and resolve inconsistencies.

4.4.1 Case description

As aforementioned, the Product and Resource Domains are already described for this case in Figure 4-6 and Figure 4-7b respectively. The extension to the Process Domain is through the introduction of new product variants, the process descriptions for which are illustrated in Figure 4-12. For Variant 1 after placing the cathode plate, first the GDL is placed and then the gasket, while for Variant 2 first the gasket is placed and then the GDL. The reason for this is some nuanced differences in the geometry of the gasket and the cathode plate which affect how seals are formed between certain components. Variant 2.1 is an extension of Variant 2 whereby an additional GDL is placed on the first one. Due to water production on the cathode side, the additional GDL serves as a tool for supporting the fluid transport.

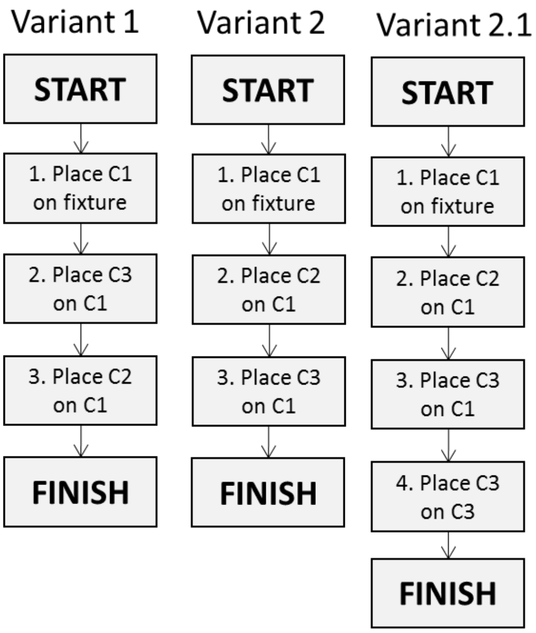
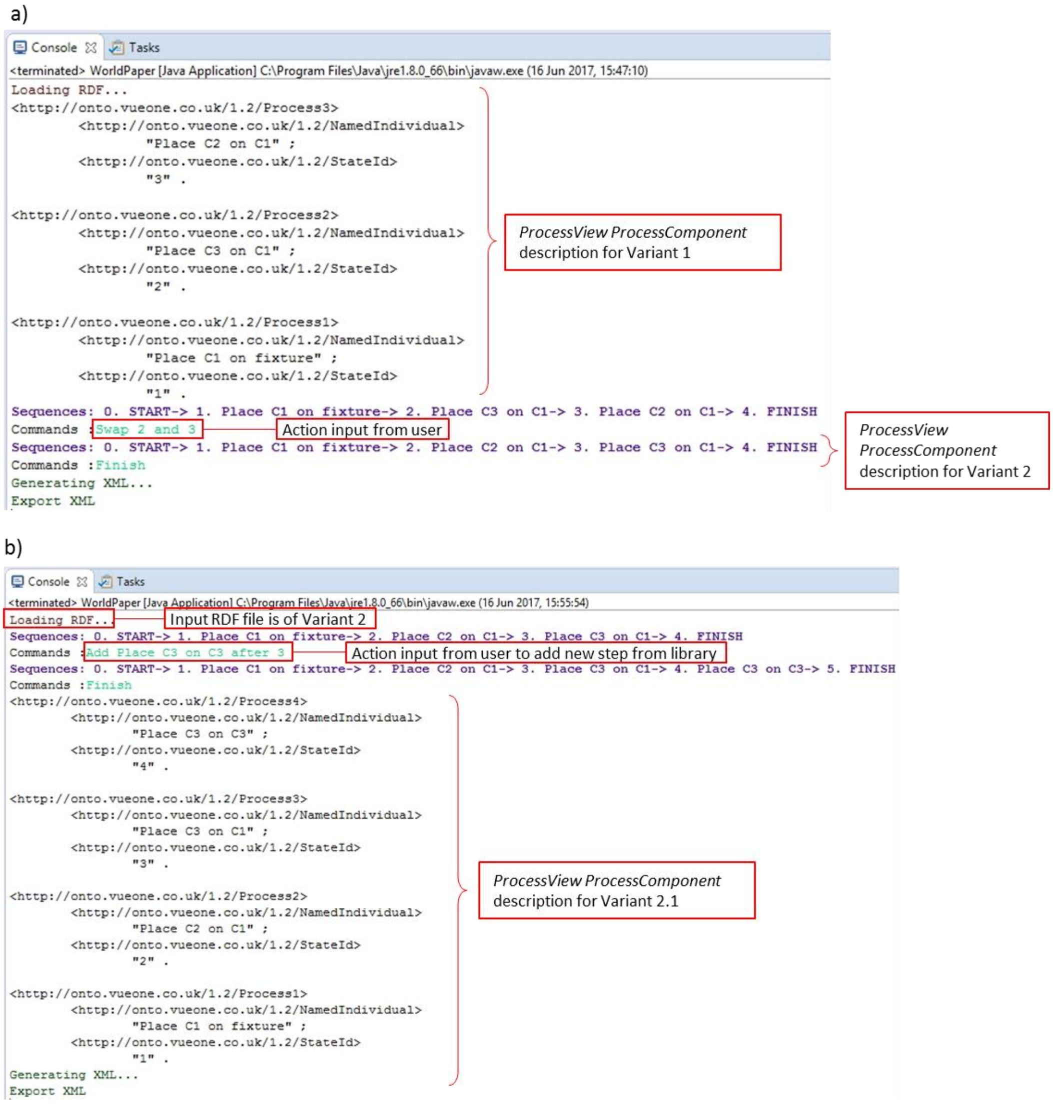


Figure 4-12 Process descriptions for additional product variants

4.4.2 Swapping and adding steps

Figure 4-13 illustrates the user interface that allows the manipulation of the ontological model so that the exported file from vueOne can be modified rapidly and then reimported for visualisation and validation purposes. In Figure 4-13a the transition from Variant 1 to Variant 2 is made by swapping steps 2 and 3 in the *ProcessView* of the *ProcessComponent*. As a consequence of doing this, through inferences via the Skill model, the low level machine logic is also modified. The change from the user perspective takes a matter of seconds needing only to type the command i.e. Swap, and then point to the steps that need to be swapped. If the equivalent change is to be made within the engineering tool prior to the introduction of the *ProcessView ProcessComponent* as well as the Skill model, a person well versed in the model takes approximately thirty minutes to make the change. On the other hand, someone familiar with the tools (but not necessarily the specific model) takes up to an hour because they first need to interrogate the model to understand the relationships between the different components and the sequence conditions. Beyond the time savings, an additional benefit is the reduction of risks as the process for making the change would normally be manually executed.

In Figure 4-13b the transformation from Variant 2 to Variant 2.1 is observed. In this case, an additional command called ‘Add’ is used. When adding a process, only those processes that exist within the library can be added. This is because this is part of the knowledge of the system as it knows that the given process is executable by the system via the Skill model. Although in this example the library is small, it is proposed that as the engineering tool library is expanded, so too are the ontologies' ability to reason available skills when adding processes. The process of adding a step in the engineering tool is more complex than swapping the steps and this is true more generally when control engineers need to insert new steps within a sequence of PLC code. This is because it is often not clear what impact the addition of a process step will have on the broader sequence as well as the risk of errors associated with not adding all necessary conditions. By providing the semi-automatic approach for adding process steps as illustrated in this section, there is both a time saving as well as a confidence that executable code can be generated. One of the shortcomings within the context of adding steps, is that the additional *ProductComponent* is not automatically added to the model and so the gantry moves without holding an object for the final step in the process for assembling Variant 2.1.



```
<terminated> WorldPaper [Java Application] C:\Program Files\Java\jre1.8.0_66\bin\javaw.exe (16 Jun 2017, 15:47:10)
Loading RDF...
<http://onto.vueone.co.uk/1.2/Process3>
        <http://onto.vueone.co.uk/1.2/NamedIndividual>
                "Place C2 on C1" ;
        <http://onto.vueone.co.uk/1.2/StateId>
                "3" .

<http://onto.vueone.co.uk/1.2/Process2>
        <http://onto.vueone.co.uk/1.2/NamedIndividual>
                "Place C3 on C1" ;
        <http://onto.vueone.co.uk/1.2/StateId>
                "2" .

<http://onto.vueone.co.uk/1.2/Process1>
        <http://onto.vueone.co.uk/1.2/NamedIndividual>
                "Place C1 on fixture" ;
        <http://onto.vueone.co.uk/1.2/StateId>
                "1" .
Sequences: 0. START-> 1. Place C1 on fixture-> 2. Place C3 on C1-> 3. Place C2 on C1-> 4. FINISH
Commands :Swap 2 and 3
Sequences: 0. START-> 1. Place C1 on fixture-> 2. Place C2 on C1-> 3. Place C3 on C1-> 4. FINISH
Commands :Finish
Generating XML...
Export XML
```

ProcessView ProcessComponent description for Variant 1

Action input from user

ProcessView ProcessComponent description for Variant 2

```
<terminated> WorldPaper [Java Application] C:\Program Files\Java\jre1.8.0_66\bin\javaw.exe (16 Jun 2017, 15:55:54)
Loading RDF..
Sequences: 0. START-> 1. Place C1 on fixture-> 2. Place C2 on C1-> 3. Place C3 on C1-> 4. FINISH
Commands :Add Place C3 on C3 after 3
Sequences: 0. START-> 1. Place C1 on fixture-> 2. Place C2 on C1-> 3. Place C3 on C1-> 4. Place C3 on C3-> 5. FINISH
Commands :Finish
<http://onto.vueone.co.uk/1.2/Process4>
        <http://onto.vueone.co.uk/1.2/NamedIndividual>
                "Place C3 on C3" ;
        <http://onto.vueone.co.uk/1.2/StateId>
                "4" .

<http://onto.vueone.co.uk/1.2/Process3>
        <http://onto.vueone.co.uk/1.2/NamedIndividual>
                "Place C3 on C1" ;
        <http://onto.vueone.co.uk/1.2/StateId>
                "3" .

<http://onto.vueone.co.uk/1.2/Process2>
        <http://onto.vueone.co.uk/1.2/NamedIndividual>
                "Place C2 on C1" ;
        <http://onto.vueone.co.uk/1.2/StateId>
                "2" .

<http://onto.vueone.co.uk/1.2/Process1>
        <http://onto.vueone.co.uk/1.2/NamedIndividual>
                "Place C1 on fixture" ;
        <http://onto.vueone.co.uk/1.2/StateId>
                "1" .
Generating XML...
Export XML
```

Input RDF file is of Variant 2

Action input from user to add new step from library

ProcessView ProcessComponent description for Variant 2.1

Figure 4-13 a) swapping process steps, and b) inserting new process step

4.4.3 Summary of Case 3

Case 3 successfully demonstrates the resolution of inconsistencies through an approach that utilises front end virtual engineering tools supported by ontologies. The approach has been demonstrated on a simple assembly which can be interrogated with relative ease by humans. However, where more complex products are involved with many components, interactions, and variants, a need arises for a method that can manage and execute changes reliably. Thus this methodology describes a more efficient way of making system changes by embedding the said expertise within a knowledge model. The key outcome from this case study can therefore be summarised as follows:

i) The correction of inconsistencies that would arise between an assembly process plan and machine control when a new product is introduced through a validation pathway via virtual engineering tools minimising risks associated with executing changes when a new process plan is generated

4.5 *Chapter Summary*

In this chapter the author has demonstrated how PPR ontologies in conjunction with the Skill model and integrated with virtual engineering tools through the Apache Jena framework, can be used to check capabilities, identify sequence inconsistencies, and resolve said inconsistencies. These case studies present a strong case for:

i) how ontological models can be used to support the engineering process in real manufacturing systems in way that has seen limited demonstration in the existing literature and

ii) substantiates the contribution to knowledge claims presented by the author in the introductory chapter

The following chapter presents an evaluation of the methodology more broadly discussing and comparing the approach with industrial state of the art i.e. PLM and existing ontologies that have been presented in the literature with similar applications.

5 Success evaluation through comparison of existing comparable works

5.1 *Introduction*

This chapter discusses and evaluates the methodology presented in this thesis. The case study chapter has demonstrated that for the given applications, the methodology is able to successfully handle the challenges. However, it is necessary to also consider the methodology within a broader context such as how it compares with similar works and what the anticipated impact or benefits would be. This forms "Descriptive Study II" of the Design Research Methodology introduced in Chapter 1. It should be noted that the *application evaluation* was carried out in the case study chapter, and the *success evaluation* is carried out here.

The core contribution of this work is the development and demonstration of modular ontological models that integrate with engineering workflows associated with product realisation. Therefore, in order to evaluate the work two comparisons with the state-of-the-art need to be made, the first being of the ontologies while the second being the framework. After this discussion this chapter summarises the evaluation, highlighting the key points.

5.2 *PPR and Skill model Ontology Evaluation*

5.2.1 Evaluation methods and criteria for ontologies

Despite the prevalence of ontological models in the literature, there is a lack of agreement as to how best to evaluate them. Hlomani and Stacey (Hlomani and Stacey, 2014) complement the definition of ontology evaluation proposed by Brank *et al.* (Brank *et al.*, 2005) as deciding the quality of an ontology, with respect to a criterion set, based on the proposed application. The definition proposed in Staab and Studer (Staab and Studer, 2010) uses the concepts of verification and validation. Ontology verification determines whether a given ontology has been built correctly while validation is concerned with identifying whether the correct ontology has been built. In this research work, the verification of the ontology has been demonstrated in the case study chapter. It can be seen that the ontology does not have any inconsistencies, includes the concepts required, and is sufficiently robust to accommodate engineering changes. Furthermore, the validation has been demonstrated in part as it is able to meet the requirements of the application cases. However, the broader validation question is whether it extends and improves upon what already exists or if it is just more of the same.

Obrst *et al.* (Obrst *et al.*, 2007) identified a need to create a systematic discipline of ontology evaluation with a view to systematically create information systems rather than the ad-hoc “close enough” approach that is prevalent in both industry and academia. They highlighted ontology evaluation techniques derived from the field of biomedicine as: application evaluation, comparing a given ontology with domain data, and performing natural language evaluations. Despite the methods described, they concluded that the best measure of an ontology is whether it has been adopted and reused. Surveys on ontology evaluation have been carried out by Brank *et al.* (Brank *et al.*, 2005) and Hlomani and Stacey (Hlomani and Stacey, 2014) and have consolidated both the methods and the criteria that exist within the literature.

The methods for ontology evaluation are summarised in Table 5-1. In most cases, the methods all suffer heavily from subjectivity or in the case of the data-driven approach a lack of appreciation of the dynamic nature of domain knowledge. However, an ontology is inherently an attempt to approximate the real world, thus the use of the term “conceptualisation” being used in all of the most highly cited definitions for ontologies. Therefore, those that create ontologies are inherently influenced by their own predilections, preferences, and expertise. As such, evaluation methodologies also suffer from the same pitfalls: a conceptualisation is being evaluated through the eyes of a person/group with their own conceptualisation. Ultimately, the method for evaluation must be able to measure the distance between the real world and the approximated conceptualisation. The challenge is determining and agreeing what the “real world” is.

To support this measurement problem, a number of metrics or criteria have been derived and some level of consensus has been reached within the literature as to what these are (Hlomani and Stacey, 2014, Bandeira *et al.*, 2016, Gómez-Pérez, 2001, Vrandečić, 2009). These criteria are summarised in Table 5-2. All of the criteria focus on the ontology, apart from “organisational fitness” which is a metric more aligned to the framework within which the ontology sits (Vrandečić, 2009).

Based on this review of methods and metrics the author proposes the following for evaluating the ontologies in this thesis:

> *Compare and contrast the PPR domain ontologies and the Skill model ontologies with “gold-standard” PPR ontologies and Skill model ontologies in the literature for:*
>
> - Adaptability

- Clarity
- Cohesion
- Completeness
- Conciseness

These metrics have been chosen because the author has sufficient information from the literature to allow an informed evaluation to be made.

The aim of this evaluation is to highlight the contribution that the ontologies in this work make and of equal importance, to ascertain the shortcomings to consider future research directions.

Table 5-1 Ontology evaluation methods

Method	**Description**
Gold standard	Comparing an ontology with a “gold-standard”. This could be an ontology generally considered to be well-structured, sufficiently expressive and complete within the domain of discourse. The key shortcoming here is the evaluation of the “gold-standard” itself, resulting in a circular evaluation problem i.e. is my actually ontology bad, or is the ontology I am comparing with bad?
Application-based	Evaluating the efficacy of an ontology within the context of an application e.g. a use case. The pitfall of this approach is that the application on which the ontology is evaluated will not be equivalent to another and thus the results cannot be confidently generalised. Furthermore, when multiple ontologies need to be compared, this can quickly become a time and resource intensive process.
Data-driven	Comparing the ontology against the existing data about the domain that the ontology is attempting to model. This can be done, for example, by comparing the ontology concepts with concepts that exist within domain documents. In this approach, domain knowledge is considered to be a constant, however this is not representative of reality where knowledge evolves as new concepts are introduced and new relations are created.
User-based	An evaluation derived from user experience. The focus is evaluating the subjective information about the ontology. The metadata from the viewpoint of the ontology creators is compared with the metadata from the view of the ontology users. This method is unable to establish objective evaluation metrics and in some cases identifying the right users can also be challenge.

Table 5-2 Quality criteria for ontology evaluation from the literature

Criteria	**Description**
Accuracy	The level of agreement between the asserted knowledge in the ontology and expert knowledge
Adaptability	Ease of use of ontology for different contexts or applications by means of extension
Clarity	The efficacy of how well the ontology communicates meaning of terms/concepts
Cohesion	A measure of ontology modularity or the level of relatedness between classes
Competency/ completeness	The coverage of a domain of interest and whether all of the necessary domains have been covered
Computational efficiency	The speed at which tools can work with the model e.g. reasoners
Conciseness	The amount of irrelevant or redundant concepts with respect to the modelled domain and thus ensuring a minimum level of ontological commitment i.e. specifying the least constraining conceptualisation
Consistency/ Coherence	The minimisation of contradictions. Also covers the consistency between formal and informal ontological representations.
Organisational fitness	Deployability of ontology for an application

5.2.2 Ontology evaluation results

The “gold-standard” ontologies used for comparing the PPR domain models are the works of Lanz (Product-Process-System model) (Lanz, 2010) and Lohse (ONTOMAS) (Lohse, 2006). Although both of these authors published their respective ontologies in journals, the author focuses only on the content as per their respective PhD theses. The reasoning for this is that the thesis would be expected to be the most comprehensive description of their ontologies including the detailed descriptions required for a comparison. Since the work of the authors cited, a number of other ontologies have been published within a similar context (see Chapter 2), however they either exist at a different level i.e. core or upper ontologies rather than domain ontologies (Lemaignan *et al.*, 2006, Borgo and Leitão, 2007, Usman *et al.*, 2013), or there is insufficient detail available about these models e.g. Hasan *et al.* and Raza and Harrison (Hasan *et al.*, 2016b, Raza and Harrison, 2011). The justification of the "gold-standard" aspect could come from the number of citations which are 16 and 43 for Lanz and Lohse respectively according to data from Google Scholar at the time of writing. As the author is unable to find other descriptive PPR ontologies in the literature aside from these, the numbers only suggest that those in this research area are aware of these works and they have been relevant enough to be cited.

On the Skill model side, the work of Järvenpää (Capability model) (Järvenpää, 2012) and Lohse (Function-Behaviour-Structure (FBS) (Lohse *et al.*, 2004) are used as the "gold standard". Although the notion of skills/capabilities have been prevalent in a number of publications and EU projects, there are limited examples of these ideas being expressed within generalizable models. Furthermore, there are even fewer examples of such models being expressed within ontologies. The works selected have explicit models that can be compared with what has been described in this work and can therefore evaluated based on the criteria selected.

5.2.2.1 Domain ontology evaluation

Table 5-3 presents the domain ontology evaluation based on the criteria selected. From the table, the key areas where the ontologies presented in this work extend what has come before are the demonstration of adaptability, cohesion, and concision from the point of view of minimising ontological commitment. With regard to the other criteria, the author believes the measures to be at least equivalent.

Table 5-3 Domain ontology evaluation

Criteria	**Evaluation**
	ONTOMAS & Product-Process-System model: No adaption of the ontology demonstrated by means of extension to include new classes in any of the case studies. However, as both works used Protégé, the ontologies could be seen as easy to use being implemented within a tool that most with the research area are familiar. Both works also created a front end for the end-user to use and interact with the ontology facilitating the exploitation of the knowledge.

	This work: Case study 1 clearly illustrated how new classes were added to the base model to allow the modelling of an assembly station that the original ontology was insufficiently expressive to support. Furthermore, general queries have been created to allow the user to exploit knowledge within the ontology. Finally, due to the integration with the Apache Jena framework a Java based GUI can be created to improve usability.
	ONTOMAS & Product-Process-System model: The words used to describe a given concepts in both works are understandable by the respective domain experts, however concept relations are not always expressed clearly. This is particularly problematic when trying to understand how domains interact with each other. The author also not find a graphic of ONTOMAS that illustrates the ontology as a whole, beyond just the very high level illustrations.
	This work: The author has ensured that the concept names are based on the "gold-standard" as well as other internationally recognised standards. In addition, the relationships between domains are expressed clearly to prevent any ambiguity.
Cohesion	**ONTOMAS & Product-Process-System model:** Both works describe the domains within ontology modules, but do not elaborate on why this is beneficial. In both models the connectedness between the Product and Resource domain low, forcing users to navigate through the Process domain for querying purposes. **This work:** As well as creating ontology modules, this work has explicitly highlighted the benefits i.e. the ability to add domains or large concepts that have not been represented. There are also connections within and across domains that facilitate knowledge exchange and interrogation.
	ONTOMAS & Product-Process-System model: Coverage of both models is comprehensive and are generally common with other manufacturing ontologies in the literature
	This work: many of the concepts have been derived from the cited works, however the shortcoming of a lack of control logic modelling has been addressed within the Resource domain
	ONTOMAS & Product-Process-System model: Not all of the concepts are used within the case study, however this may well be because of the nature of the cases themselves and thus cannot be fully attributed to redundant concepts. ONTOMAS is based on a core ontology and therefore a level of ontological commitment is imposed.

	However, in both cases there are no highly constraining axioms that unduly constrain the ontology
	This work: As with the cited works, not all of the concepts have been used, but this is attributed to the nature of the case studies. The ontology offers minimal ontological commitment within a number of classes e.g. that a *ResourceComponent* could be an instance of *SkillContext.* Preventing an unnecessary constraints upon the ontology has been key to ensure that a myriad of engineering workflows can be accommodated which the ultimate aim of the work.

5.2.2.2 Skill model evaluation

The Skill models being compared in this section of the evaluation are given in Figure 5-1, Figure 5-2, and Figure 5-3 for the Capability model, FBS model, and the Skill model in this work respectively. Note that Figure 5-1 and Figure 5-3 already exist in the literature review chapter and the methodology chapter respectively, but have also been included here to ease the evaluation process.

Table 5-4 presents the Skill model evaluation based on the criteria selected. From the table, the key areas of contribution are the cohesion, completeness, and concision. The main shortcoming of the Skill model is the clarity as compared to previous works, but in other respects the model is at least equivalent.

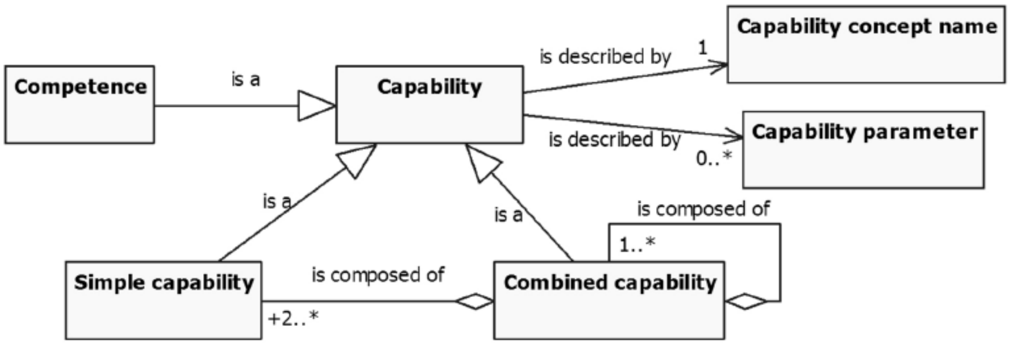


Figure 5-1 Capability model (Järvenpää, 2012)

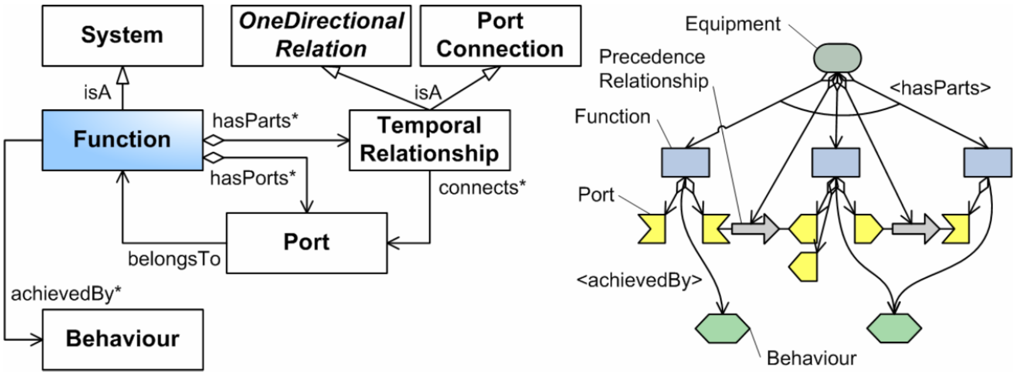


Figure 5-2 FBS (Lohse et al., 2004)

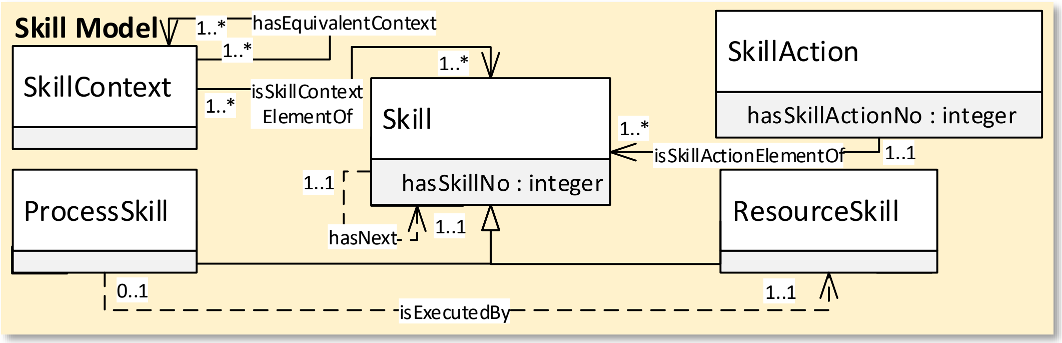


Figure 5-3 Skill model (this work)

Table 5-4 Skill model evaluation

Criteria	**Evaluation**
	Capability model & FBS: The Capability model is inherently adaptable due to its ability to aggregate capabilities into new ones allowing the ontology to evolve. FBS is more rigid due to the direct link between *Function* and *Equipment*.
	Skill model: In the same vein as the Capability model, the Skill model can model both atomic Skills through *SkillAction*, but also their aggregation through the *Skill* class. The usability is improved as the granularity of an instance of *Skill* is user-dependant and dictated in part by the requirements of the Process domain.
	Capability model & FBS: The terms used in the Capability model are clear and even a non-expert would be able to grasp what is being modelled. The terms in the FBS model and their relationships are more abstract due to constraints from the core ontology. The user would need

	to scrutinise documentation associated with the FBS model to understand what is being represented.
	Skill model: The terms used are not especially clear and the author does feel that the user would need to consult documentation to understand what is being modelled. However, as the Skill model forms part of a broader framework and would be used via front end engineering tools, it is not believed to be especially concerning
Cohesion	**Capability model & FBS:** The Capability model exists solely within the Resource domain and it is not clear from the work how it interacts with other concepts. There is a greater level of cohesion in the FBS model, attributed to the core ontology. However, in both cases, the models are part of the Resource domain models and therefore limited links with other concepts exist. **Skill model:** There are a number of interactions with all of the PPR domains demonstrating a more integrated model that is able to take information from multiple sources and reason accordingly.
	Capability model & FBS: Both models are complete in an absolute sense, with the Capability model including the human factor of competence which is missing in FBS. On the other hand, FBS appreciates the need to model the sequence modelled through temporal relationships which are missing in the Capability model.
	Skill model: The model extends the absolute nature of skills or capabilities that exist within the literature and contextualises increasing the expressivity and thus reasoning ability. The temporal nature of skills are modelled through data type properties, while human competences would be handled through an action carried out by a specific instance of a *ResourceComponent* i.e. a human
	Capability model & FBS: The Capability model requires only 6 classes for an expressive model. The FBS illustrated in Figure 5-2 is the highest level version of the model and so there are a number of additional concepts. While this may aid expressivity, it could affect the usability of the model require a large amount of information for instantiation.
	Skill model: There are 5 main classes in total but regardless of this the model is more expressive than the Capability model. All of the classes have been used in the case study and so a lack of redundancy has been demonstrated.

5.2.3 Discussion of ontology evaluation

One of the limitations of the evaluation is that it is almost impossible to find any work that has identical aims and objectives. Therefore, it must be stressed that an identified shortcoming of a given work may exist simply because it was not within its scope and is not intended to demean it in any way.

One of the important aspects of this work is that the ontologies for the PPR domains and the Skill model are expressive and declarative about their relationships. This ensures that the user can understand the rationale behind the link and then create queries that follow a more natural logic than would be the case within an SQL database where the schema is hidden. Furthermore, previous works within the same area e.g. Lohse(Lohse, 2006), Lanz (Lanz, 2010), and Järvenpää (Järvenpää, 2012), have looked at either the PPR ontology or the Skill model individually, but not brought them together under one integrated model. These works have also not made it clear how and where cross domain links are formed. This is of paramount importance as it at these domain interfaces where information is lost and ensuring that clear links exist allows the user to identify why certain pieces of information may not be carried across domain i.e. due to a lack of expressivity of a given domain concept.

The author has also been keen to maintain a modularised approach to the ontological models. Rather than all of the ontologies existing in a single file, the respective models are independent from each other. This highlights the importance of declaring how the respective domains are linked. Moreover, due to the modularised approach, sub-ontologies can be added to accommodate concepts or domains that have not been included in this work. For example, one of the important aspects of engineering change is weighing up the cost of different options vs. the benefit they would provide. Therefore, a cost-model ontology could be attached as a module, interacting with the relevant concepts from within the respective domains to predict the investment required to implement a given solution.

Furthermore, the approach lends itself to the standardisation of terminology e.g. the *SkillAction* class via DIN 8580 (DIN, 2003) or VDI 2860 (VDI, 1990), so that it can be used across a host of applications. It also acknowledges efforts to move towards common automation system lifecycle semantics through efforts such as AutomationML (AML) (Drath *et al.*, 2008), and thus provides a “semantic exchange” layer that enables those users that would prefer its notation to do so. The work presented in Kovalenko *et al.* (Kovalenko *et al.*, 2015) experimented with comparing a model-driven approach (Ecore) with a semantic web approach (OWL) for representing AML. The author raises this work here to highlight that such projects evidence the rising appreciation for semantics within relevant exchange languages. In the cited work, it was identified that the transformation of AML to OWL was challenging and required several iterations. This could potentially have been resolved by mapping the necessary concepts within AML to the PPR ontology presented in this research work.

One of the fundamental tenets of ontological models is their ability to represent knowledge. However, knowledge is inherently dynamic and fluid and the scientific community is an excellent example of this. As new findings are made, updates are made to the body of knowledge. Within the field of knowledge representation, this fluidity has been recognised and addressed through dynamic knowledge representation (Alferes *et al.*, 2000a, Javed *et al.*, 2013). There is a significant body of literature in this area and much like this research it focuses on updating knowledge models and managing conflicts. In Alferes *et al.* (Alferes *et al.*, 2000b) a “Language of UPdateS” (LUPS) was created that, among other things, allowed the authors to store and thus query the history of the knowledge models. The importance of this ability was exemplified through legal reasoning where the law evolves but knowledge about the state of the law prior to the current time is necessary to know and thus determine whether a crime was committed in the past. Dynamic ontology evolution was explored in Zablith (Zablith, 2008) through a framework called Evolva. The framework consisted of several steps, namely: information discovery, data validation, ontological changes, evolution validation, and evolution management. Of note was the evolution management as recorded the changes made to the ontology to allow changes to be rolled back. In Heflin and Hendler (Heflin and Hendler, 2000) the problems associated with managing ontologies within distributed environments (akin to the design and engineering activities associated with product realisation) and addressed through SHOE (Simple HTML Ontology Extensions). They developed three ontology integration methods which were:

1. Mapping ontologies: to assimilate different ontologies into a single one
2. Mapping revisions: to use rules to update multiple ontologies based on the updates within one or many
3. Intersection ontology: where a new ontology intersects the concepts between pre-existing ontologies, a process of renaming terms is carried out

A combination of these methods could be relevant to ontologies that are used within manufacturing. The modular nature of ontologies presented in this work is a double-edged sword, while it reduces model complexity and facilitates integration, there is a greater risk of multiple versions evolving and diverging. Therefore it is important to implement such methods to prevent inconsistencies and divergences arising in a model which is relied upon to be consistent and the “single source of truth”.

The dynamic nature of knowledge in manufacturing system and the need to be able to query its history is also important. However, within the approach presented in this research, the historical aspect of the system was out of the scope. The author believes that it is important to include this in future work so that i) a previous version of the manufacturing system can be interrogated for its skills to find out for example how a previous product variant was assembled, and ii) ascertain how the system evolved so that the system can be reverted to its original state (particularly useful for control code) if there are issues with the current version.

5.3 *Framework evaluation*

5.3.1 Framework evaluation approach

The ontologies developed in this research were developed, from the outset, to sit within a framework envisioned to bring together knowledge representation with engineering tools, methods, and workflows. This was with a view to addressing a question as to how ontologies can support new product introduction to combat reduced product lifecycles and an increased number of product variants. The framework created formed a hierarchy that included the multiple levels that exist within industry e.g. users and tools, and complemented them with a knowledge level. The knowledge level was represented using a PPR ontology in conjunction with an innovative, extendable Skill model.

This section of the evaluation compares frameworks that have been developed which are similar in their scope as the one created in this work. This section does not use the more formal evaluation process presented to evaluate the ontological models. Despite a literature survey, the author has not found a methodology for evaluating architectures and frameworks. As such, the author describes how well the frameworks address the following goals:

1. The framework should clearly describe how the ontological models fit and the value knowledge representation brings
2. The framework should show how it facilitates the engineering workflow through change management or the given engineering process it was designed to support

3. The framework should show an appreciation for user interaction and how this is accomplished i.e. through integration with engineering tools

5.3.2 Comparable frameworks

Ultimately the framework should be demonstrating a value-adding industrial case for utilising knowledge representation. The frameworks that have been selected have already been discussed in the literature review chapter, however they are discussed in more detail here and with the perspective of facilitating the evaluation of the author's research. The works selected to compare and contrast for this evaluation are as follows:

1. **Alsafi and Vyatkin** (Alsafi and Vyatkin, 2010) where the objective was to utilise knowledge representation to achieve fast reconfiguration of modular manufacturing systems through an ontology agent. The agent inferred facts about the manufacturing environment from the ontological model and determined whether it was capable and then derive new configurations. The layered architecture of the ontology-based reconfiguration architecture is presented in Figure 5-4. The specifications layer explicitly provides all the raw data about the manufacturing system including the requirements e.g. process information, the layout, and the knowledge about the system itself. Note that the knowledge is represented within an OWL-DL file. The analysing and modelling layer interprets the information from the specifications layer into models that can be accessed and manipulated. The intelligent reasoning layer reasons about the requirements, and based on the layout and capabilities generates a final configuration. The deployment manager in this top layer is mostly focused on deploying software changes associated with the logical operation of the manufacturing system. In order for the deployment manager to work, the caveat is that the low level distributed system controllers are in compliance with IEC 61499 (Vyatkin, 2009).

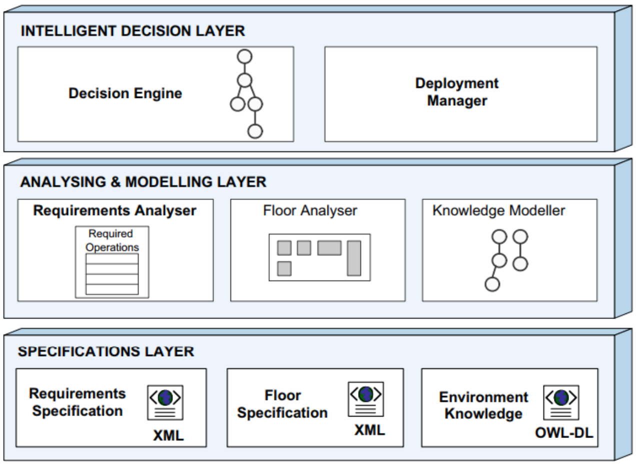


Figure 5-4 Layered architecture of ontology-based reconfiguration agent (Alsafi and Vyatkin, 2010)

2. **Lanz** (Lanz, 2010) integrated the Product-Process-System model ontology into a broader conceptual architecture that is illustrated in Figure 5-5. There are four access layers which create software modules with their own responsibilities. Information is implemented within XML-based files with the service interface layer allowing the client layer access to the knowledge stored in the knowledge base. The service layer serves as an access layer between the respective mappers for the different formats i.e. X3D, VRML etc. These mappers facilitate data exchange between different web applications. The ontology manager in the service layer is responsible for modifying the content of the ontology based on updates from the client layer data. The ontology access layer serves to provide reasoning (via Pellet reasoner (Sirin *et al.*, 2007)) and conflict avoidance capabilities due to the distributed nature of the approach from the user perspective.

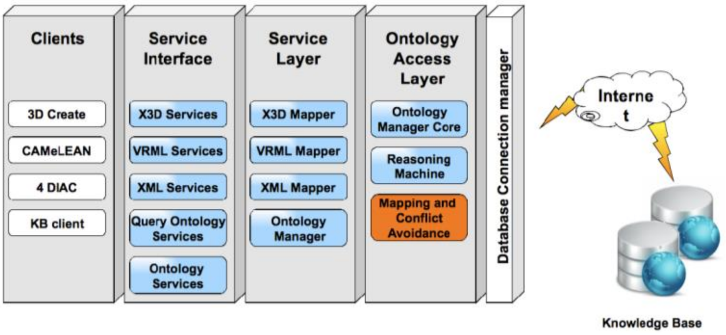


Figure 5-5 Conceptual architecture of knowledge base (Lanz, 2010)

Finally in Figure 5-6, the framework proposed and demonstrated in this research is illustrated. This has already been described in detail in Chapter 3, however the figure has been reproduced to facilitate easier reference for the reader.

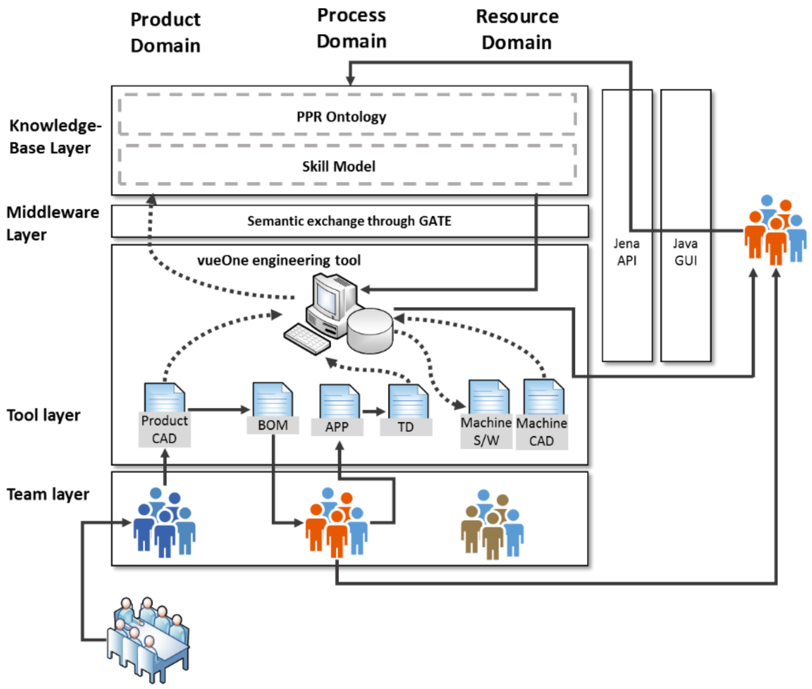


Figure 5-6 Framework of this research

5.3.3 Framework evaluation results

5.3.3.1 Fit and value of KR in frameworks

Both of the frameworks in the literature demonstrate and clearly describe the benefits of using ontologies in their works and are also clear in describing how they are used. However, Alsafi and Vyatkin (Alsafi and Vyatkin, 2010) the specification layer includes both XML and OWL files, while Lanz (Lanz, 2010) attempts to keep ontological models separate to XML files with them interacting through mappers. In this work, the framework shares some similarities with Lanz (Lanz, 2010) with respect to the fit of the ontologies, and sees the work in Alsafi and Vyatkin (Alsafi and Vyatkin, 2010) to be a significantly narrower in vision and thus reducing the value of implementing the ontology within the framework. The framework significantly extends what has been proposed by implementing a Semantic Exchange layer which is not limited by a specific type of data format (although XML is used to demonstrate it) as is the case with the other frameworks. As aforementioned the consideration for broader standards to integrate the tools with the ontology leads to a more generally usable and thus valuable framework.

5.3.3.2 Contribution of the framework to the engineering workflow

Alsafi and Vyatkin (Alsafi and Vyatkin, 2010) contributed to the workflow at the reconfiguration phase of a manufacturing system while Lanz (Lanz, 2010) was more focused on knowledge retrieval which could exist at any phase. Due to the deployable nature of the work carried out by Alsafi and Vyatkin (Alsafi and Vyatkin, 2010) there was more value added as a consequence of using the framework as compared to Lanz (Lanz, 2010), despite its narrower application. This work demonstrates a framework that contributes strongly through multiple lifecycle phases across multiple domains and considers both the existing engineering workflow and the way that the generated knowledge is managed in a way that complements industrial practices. The framework allows the retrieval of knowledge, ascertaining the consistency across domain models, and also rectifying such inconsistencies through parametric or logical changes. This is demonstrated by the queries in Case Study 2 that appreciate the nature of the information that would need to be queried at a given lifecycle phase, within a given domain, and how information would flow through the ontology in a way that mimics the human interrogation processes.

This section of the evaluation therefore surmises that an extension to what has come before with respect to frameworks to support the engineering workflow with ontologies, has been achieved through the research in this thesis.

5.3.3.3 User interaction with the framework

The framework presented by Alsafi and Vyatkin (Alsafi and Vyatkin, 2010) did not explain how users would interact with the model with the assumption being that the entire process would be automated. Lanz (Lanz, 2010) hints at some level of user interaction by mentioning tools that are integrated with the framework. However, in the framework presented in this thesis, there is a substantially clearer demonstration of how and why users would interact with the framework and the ontology more specifically.

In addition, due to the integration with a component-based virtual engineering environment, changes can be assessed before being implemented. The “component-based” element is important to recognise as it does not reflect the industry standard for engineering software. This prevents industrial engineering tools from being extensible and thus preventing a number of key concepts represented within this approach, particularly the Skill model, from being implemented directly into conventional engineering tools. On the other hand, the vueOne toolset can be extended and the respective Skills form an attribute of the different component types, be they *ProductComponent, ProcessComponent,* or *ResourceComponent.* Therefore, to realise Objective 3 it was identified that the framework developed cannot benefit industrial needs unless either software vendors embrace a more open approach and reveal the nuances of their models, or a significant effort is undertaken to understand the data models and map the concepts to neutral exchange standards such as AML. The framework accommodates both scenarios, but does rely on the work of future researchers for full implementation.

The “virtual engineering environment” integration facilitated a “zero-risk” nature of experimenting with changes. This was not present in Alsafi and Vyatkin’s (Alsafi and Vyatkin, 2010) work where the intention appeared to be direct deployment of modifications with the assumption that risks had been mitigated through reasoners. However, in the framework in this thesis, steps and logic can be tested without affecting the real system until necessary. Although existing industrial virtual engineering tools for system modelling offer the ability to model changes, the prerequisite remains that the user is operating within the Resource domain i.e. there is no link with high level process descriptions. As a result, changes within one domain must be transformed through domains by human intervention, this can lead to errors as a result of miscommunication. Thus, this work demonstrates how ontological models can be “used” within practical engineering workflows.

5.3.3.4 Additional remarks

5.3.3.4.1 Relevance to PLM

In the literature review chapter, the shortcomings of PLM was identified and it was noted that the research work in this thesis would need to compare with such solutions. The conventional PLM tool chain has suffered from information loss as the lifecycle progresses. Furthermore, there is (as the name suggests) a heavy focus on product information within this paradigm. As a result, the respective tools and methods that have emerged as an outcome of attempting to align themselves within this paradigm have taken to a similar design philosophy. Tools that the author would class within the Resource domain lack the expressivity required and typically do not integrate well with other Resource domain tools. On the other hand they do retain a substantial amount of Product domain information e.g. geometry, material characteristics (Demoly *et al.*, 2011, Lee *et al.*, 2011b) etc. The significance of this within the context of this research is the PLM paradigm and the PPR approach are not aligned. This means that should an integration framework of the nature presented in this work be successful for an industrial application, it would be necessary for an alignment procedure to be carried out and some efforts to bring PLM and PPR together. This challenge has been identified by a number of academics and there are ongoing projects that aim to facilitate the exchange of data between PLM and PPR to address this (El Kadiri and Kiritsis, 2015, Matsokis and Kiritsis, 2010, Milicic *et al.*, 2013, Choi *et al.*, 2010).

5.3.3.4.2 Concerns with OWL

The globalisation of manufacturing enterprises means that those designing the product do not sit in the same geographical location as those considering the process, nor those that design and commission the system. Furthermore, the manufacturing system itself may well be in a different country. The exchangeability of data formats such as XML in conjunction with semantic web technologies present a solution to this problem because OWL models can be published on the World Wide Web thus providing access to anyone with a web browser. This is not typically the case for engineering tools and thus supports more distributed engineering activities. However, OWL has seen limited implementation within industrial settings, particularly in manufacturing, despite its robustness. This has been proved in a number of works, particularly in large EU funded research projects that are summarised in Chapter 2. Despite the strength of the language, the tools used for implementation e.g. Protégé, remain largely the plaything of academics. Furthermore OWL 2, which is the most recent version of OWL, was published in 2012. The language has not had the time to proliferate through the education system and thus there is a lack of expertise to realise implementation. Thus, despite the benefits of the approach demonstrated in this work, the move into an industrial environment is hampered by a lack of expertise and tools.

5.4 *Summary*

The evaluation of any piece of research is fundamental in determining whether or not any novelty exists and there is a significant contribution to the body of knowledge. In this chapter, the author has evaluated the ontology through metrics and a method derived from the literature, and the broader framework based on the objectives of this thesis. An application evaluation was carried out in the preceding chapter based on case studies.

The key points from the ontology evaluation are a more adaptable, cohesive, concise, and complete model than has been previously presented in the literature that brings together domain models with an independent Skill model spanning the domains. From the framework evaluation the author identifies that a greater level of value can be derived as compared to previous similar frameworks based on its broader scope and usability.

The following chapter concludes the work and based on the gaps and shortcomings identified proposes future steps.

6 Conclusion and Further Work

6.1 *Introduction*

As the paradigms of mass customisation and product personalisation become ever prevalent, it is clear that the challenges facing the industry today are accommodating these dynamic market conditions while maintaining profit margins and productivity. At the highest level the questions were to understand why making changes was problematic, and why existing methods for representing knowledge were not addressing the needs of the industry? Following this, the author wanted to address what knowledge models and broader frameworks should look like to support the uncertainty facing manufacturers today. To address these issues, the objectives of the thesis were as follows:

1. Identify change management methods within the context of manufacturing and engineering changes and the challenges that are faced
2. Identify the ontological models that have been developed in the literature and how they have been applied as well as their shortcomings
3. Develop a set of PPR ontologies that can be used to support assembly automation systems engineering through its lifecycle
4. Develop a framework that integrates engineering tools, methods, and workflows with an ontological model
5. Demonstrate how ontologies can be used in a practical way to identify and resolve inconsistencies

These objectives can be classed into two categories. The first category is for objectives 1 and 2 and is the identification of knowledge gaps and shortcomings of existing works that address the same problems. The second category is for objectives 3, 4, and 5 and define the key contributions of this work.

6.2 *Summary of knowledge gaps*

6.2.1 Objective 1

With regards to **Objective 1**, the literature review identified that engineering changes are the source of significant costs due to complex workflows attributed to multiple engineering activities, product realisation domains, and domain specific languages and models. The notion of minimising engineering changes, grouping them together, or avoiding the process altogether is deemed to be somewhat archaic by the author, although works that aligned with this philosophy were found to exist within the literature. In a sense the paradigm of flexible manufacturing systems (FMS) aligns with this notion as such systems are designed to be changeable within the limits of the system itself, which if considered from the perspective of the system, is not changeable at all. The position the author took (which aligns with the paradigm of adaptability in manufacturing (Keddis *et al.*, 2013, Keddis *et al.*, 2014)), and thus formed the motivation for the research, is that change should be encouraged, supported, and undertaken with gusto. However, it was confirmed from the literature review that the bureaucracy and administration associated with making such changes caused a significant level of apprehension for those involved.

6.2.2 Objective 2

Resolving **Objective 2** was also a process carried out in the literature review chapter. It identified that despite the prevalence of knowledge representation through ontologies within the literature, the connection to software tools was limited, and lacked the expressivity required to be of significant practical use for industrial applications.

6.3 *Key Contributions*

The key contributions of this thesis are summarised in this section with references to the sections in the thesis where these contributions have been made or evidence to that effect.

6.3.1 Objective 3

As a consequence of the knowledge gaps identified, the author first focused on what concepts should exist within the respective PPR ontologies and furthermore to create a reusable, extensible, and integrated Skill model. This was seen to be a key enabler of more complex queries and thus bringing together the industrial engineering workflow associated with realising automation systems with knowledge representation. As such, the first key contribution of this research thesis as justified by both the application evaluation in the case study chapter, and success evaluation in the previous chapter is the:

- **Development of adaptable, cohesive, concise, and complete PPR Ontologies and Skill Model to support the storage and reuse of knowledge within the context of industrial automation systems.** There are many examples of manufacturing ontologies in the literature. However, only a limited number model the skills or capabilities of the manufacturing system and consider how they map to the requirements of the product or the process. Rather than simply presenting a hierarchy of concepts within the Resource domain, it is necessary to contextualise the information so that it is useful for domain stakeholders. This contribution has been achieved by

a. Identifying the shortcomings of ontological models that have been generated within the area of manufacturing (*Section 2.6.1*)
b. Developing new ontological models that focus on assembly by using pre-existing models and extending them (*Section 3.4.4, 3.5.2, 3.6.3*)
c. Introducing a Skill model that integrates the respective PPR ontologies to contextualise information, manage inconsistencies, and add more value to the engineering workflow (*Section 3.7.1*)
d. Demonstrating extensibility of ontological concepts within the ontology without resulting in inconsistencies and contradictions (*Section 4.2.3, Figure 4-2*)

6.3.2 Objective 4

The existence of models that represent knowledge are, on their own, not able to support industrial needs. The knowledge gap addressed in this case is therefore identifying the nature of the framework that needs to exist to holistically support industrial automation system engineering from the product design through to control code generation. Therefore, the second key contribution of this work is the:

- **Development of a framework that brings together the PPR ontologies and Skill model with existing engineering tools and the associated domain stakeholders.** This has been achieved and demonstrated through evaluation by:
    a. Significantly extending previous frameworks of a similar nature by eliminating the limitation of working solely with XML files by introducing a flexible semantic exchange layer, that could be supported through standards that are growing in popularity e.g. AutomationML (Introduced in *Section 3.8.3, Figure 3-20* and Evaluated in *Section 5.3.3*)
    b. Supporting existing engineering workflows more broadly than previous works have through general queries (*Section 3.8.1, Figure 3-15, 3-16)*
    c. Providing an access point for stakeholders that may be non-experts within the context of ontology models through the Apache Jena framework which sits in a position where it can interact both with source models as well as the knowledge models (Introduced in *Section 3.8.3, Figure 3-20* and evaluated through proof of concept demonstration in *4.4.2*)

    d. Extending the data model of the vueOne engineering toolset to accommodate additional concepts that previously have not existed with a view to presenting a set of recommendations for software vendors creating tools within similar domains (*Section 3.6.2, Table 3-3*)

6.3.3 Objective 5

One of the problems identified that the author wished to resolve through knowledge representation was identifying and resolving inconsistencies that arise between models that exist in different domains, using different language, and are expressed at different levels of granularity. The focus therefore turned to how process plan models can be mapped to control code with a view to maintaining logical consistency. Therefore the third key contribution of the work was:

- **A method for mapping process plans at different levels of abstraction with control code through the novel Skill mode to identify and resolve inconsistencies, facilitated by visualisation and verification through virtual engineering tools.** Although there have been some works that identify how inconsistencies can be resolved across models, addressing and accommodating the multiple and varying granularities of different domain models has not been resolved. This contribution has been achieve by:
    a. Exploiting the ontologies and the framework presented to support and partially automate the engineering workflow associated with maintaining the consistency between process plans and machine control code (Introduced in *Section 3.8.2*, *Figure 3-18* and evaluated through proof of concept demonstration in *4.4.2*)

In summary, the thesis identifies the shortcomings of existing ontological models within the context of manufacturing, develops new models to address those shortcoming, and develops new, useful ways for ontological models to be used to address industrial problems by integrating them with virtual engineering tools. By formulating a method that can integrate the assembly process sequence changes to machine control logic in a way that facilitates visualisation and ultimately commissioning an important step to realise practical engineering concurrency in industrial automation has been achieved.

Reflecting upon the original hypothesis of this thesis that:

"*Ontologies can be integrated with engineering tools to complement existing engineering workflows through the identification and resolution of inconsistencies between typically* *un-integrated and disparate engineering models, complementing and enhancing the capability of databases"*

the following conclusions are derived:

1. Ontologies can be, given a suitable framework, integrated with exisiting engineering tools and workflows as illustrated in this thesis to address issues associated with resolving inconsistencies across models. However, there remains the challenge of adopting a new, unfamiliar technology that requires a workforce skilled in its use should ontologies be moved into an industrial setting.
2. The use of semantics and inference allows the enhancement and exploitation of data that exists within databases. However, what is called into question, particularly within an industrial setting, is the performance of ontological models in comparison with databases and this must be investigated as part of future work.

6.4 *Further Work*

Although this research has successfully and comprehensively addressed the objectives of the thesis, a number of new questions and problems have been raised. These are discussed in this section.

6.4.1 Supporting more complex process logic changes

The presented algorithm in Case Study 3 had the capability to swap, add, and remove sequence steps from the process. One of the limitations that has been mentioned is the lack of ability of the model, and thus the algorithm, to accommodate branched and parallel processes. One of the reasons for this is the use of state numbers as a means for checking consistency between the *ProcessView* and *ResourceView ProcessComponent* sequence. Essentially, when mapping the *ProcessSkill* and the *ResourceSkill* the *hasStateNo* datatype property is used as a means for ensuring sequence consistency. However, when considering branched and parallel processes, the sequence is not linear. This results in state numbers that are not sequential as a result consistency cannot be ascertained using this methodology. This problem is exacerbated by the lack of a standardised way to number branched and parallel logic. In future work, the consistency check model is to be extended which in turn is expected to allow the capabilities of the algorithm presented in this work to be extended also. More formal semantics could be used to enable the approach to work off of reasoning from the language used by the respective domains when describing process steps.

6.4.2 History management

The evaluation chapter discussed the importance of managing historical knowledge to allow a knowledge model to revert and thus query a previous state of existence. Implementation of this could be supported through methods that have developed in the literature. One of the interesting questions that could be answered if such history was implemented on a wider scale is surrounding how manufacturing systems evolve and thus whether any predictions can be made about the future.

6.4.3 Full implementation with virtual engineering tools

Although some level of implementation was achieved in this research, there remained a manual process of importing and exporting files. In addition, further validation work is required before the proposed approach can be offered to industry for implementation. While this approach was sufficient to prove the validity of the method within an academic setting, future work needs to fully implement the envisioned framework. As a consequence, the challenges that cannot be seen at the more conceptual stage will be uncovered and addressed resulting in further validation. The full implementation would also allow the case studies that have been explored within a proof-of-concept level demonstration to be tested in a more prototypical, stable environment so that engineers to use the system with some level of autonomy. This would provide feedback that would in turn result in further development and refinement of the workflow. The key outcome of this work would be to evaluate to which degree such an approach mitigates risk associated with engineering changes, which formed the primary motivation for this work.

6.4.4 Mechanical reconfiguration

The software aspect of reconfiguration as consequence of new requirements was addressed in this research. However, there is also the physical, mechanical nature of change that needs to be supported. Ontological models are not well-suited to representing complex geometrical information. Therefore, the author proposes that work needs to be done within virtual engineering tools themselves to support mechanical reconfiguration supported by ontologies that advise the aspects that may need to be changed. The author is engaged in developing this idea in collaboration with colleagues at the Technical University of Munich, Germany. The vision is to develop an algorithm that integrates with the vueOne toolset that can determine what steps need to be taken to mechanically reconfigure a machine based on its existing state and new product requirements.

6.4.5 Fuel cell manufacturing knowledge

During the course of the research project, the author engaged heavily with a number of fuel cell research projects focused on manufacturing and assembly. The author hoped to fully integrate the fuel cell knowledge into the ontological models within the time-frame of the PhD research as a referenceable knowledge model that can be used by fuel cell manufacturers that are expected to emerge as the technology matures. It would further validate the work and also extend the model with concepts that do not currently exist. The author intends to carry out this work in due course with the permission of industrial collaborators.

6.4.6 Web implementation

The globalisation of manufacturing enterprises means that those designing the product do not sit in the same geographical location as those considering the process, nor those that design and commission the system. Furthermore, the manufacturing system itself may well be in a different country. The exchangeability of data formats such as XML in conjunction with semantic web technologies present a solution to this problem because OWL models can be published on the World Wide Web thus providing access to anyone with a web browser. However, this was not implemented in this research work due to time constraints. Demonstration of this would have presented a stronger case for modular ontologies but also required serious consideration for factors such as access control and security.

References

ATG Simulation Approach [Online]. Available: http://www.atg.gb.com/simulation/ [Accessed 2016].

Definitions for Product, Process, and Resource [Online]. Available: http://www.businessdictionary.com/ [Accessed].

Delmia 3D digital manufacturing solution, Dassault Systems [Online]. Available: http://www.3ds.com/products-services/delmia/ [Accessed].

GATE - General Architecture for Text Engineering [Online]. Available: https://gate.ac.uk/ [Accessed 2016].

IDEAS - Instantly Deployable Evolvable Assembly Systems [Online]. Available: http://www.ideas-project.eu/ [Accessed].

ontology editor developed by Stanford University [Online]. Available: http://protege.stanford.edu/ [Accessed].

RDF Working Group, "Resource Description Framework (RDF) [Online]. Available: http://www.w3.org/RDF/ [Accessed 2016].

Siemens Teamcenter [Online]. Available: https://www.plm.automation.siemens.com/en_us/products/teamcenter/ [Accessed].

SimSol - Simulation Solutions [Online]. Available: http://www.simsol.co.uk/ [Accessed 2016].

SolidWorks - 3D software tools. *Dassault Systemes*.

Tata Consultancy Services - Digital Software and Solutions Group [Online]. Available: http://www.tcs.com/digital-software-solutions/pages/default.aspx [Accessed 2016].

Visual Components [Online]. Available: http://www.visualcomponents.com/ [Accessed 2016].

W3C Recommendations - RDF Current Status [Online]. Available: http://www.w3.org/standards/techs/rdf#w3c_all [Accessed 2016].

2004. Configuration Management Plans - From Traditional CM to CMII (Rev B), White Paper.

2008. SIARAS - Skill-based Inspection and Assembly for Reconfigurable Automation Systems

2010. *Module 13 - Introduction to Semantic Technology, Ontologies and the Semantic Web* [Online]. GATE. Available: https://gate.ac.uk/wiki/TrainingCourseAug2010/track4-slides/module-13.pdf [Accessed June 2017].

ADOLPHS, P., BEDENBENDER, H., DIRZUS, D., EHLICH, M., EPPLE, U., HANKEL, M., HEIDEL, R., HOFFMEISTER, M., HUHLE, H. & KÄRCHER, B. 2015. Reference architecture model industrie 4.0 (rami4. 0). *VDI/VDE Society Measurement and Automatic Control (GMA)*.

AHMAD, B. 2014. *A component-based virtual engineering approach to PLC code generation for automation systems.* Loughborough University.

AHMAD, M., AHMAD, B., HARRISON, R., FERRER, B. R., LASTRA, J. L. M., MEREDITH, J. & BINDEL, A. A knowledge-based approach for the selection of assembly equipment based on fuel cell component characteristics. Industrial Electronics Society, IECON 2015-41st Annual Conference of the IEEE, 2015a. IEEE, 001002-001007.

AHMAD, M., AHMAD, B., HARRISON, R., RAMIS FERRER, B., LASTRA, J. M., MEREDITH, J. & BINDEL, A. A knowledge-based approach for the selection of assembly equipment based on fuel cell component characteristics. IECON, Nov. 9-12, 2015 2015b Yokohama, Japan. IEEE.

AHMAD, M., ALKAN, B., AHMAD, B., VERA, D., HARRISON, R., MEREDITH, J. O. & BINDEL, A. 2016. The use of a complexity model to facilitate in the selection of a fuel cell assembly sequence. *Procedia CIRP***,** 1-6.

ALEKSANDROV, K., SCHUBERT, V. & OVTCHAROVA, J. Skill-Based Asset Management: A PLM-Approach for Reconfigurable Production Systems. IFIP International Conference on Product Lifecycle Management, 2014. Springer, 465-474.

ALFERES, J. J., PEREIRA, L. M., PRZYMUSINSKA, H., PRZYMUSINSKI, T. C. & QUARESMA, P. 2000a. Dynamic Knowledge Representation and Its Applications. *Artificial Intelligence: Methodology, Systems, and Applications: 9th International Conference, AIMSA 2000 Varna, Bulgaria, September 20–23, 2000 Proceedings.* Berlin, Heidelberg: Springer Berlin Heidelberg.

ALFERES, J. J., PEREIRA, L. M., PRZYMUSINSKA, H., PRZYMUSINSKI, T. C. & QUARESMA, P. 2000b. An exercise with dynamic knowledge representation. December.

ALSAFI, Y. & VYATKIN, V. 2010. Ontology-based reconfiguration agent for intelligent mechatronic systems in flexible manufacturing. *Robotics and Computer-Integrated Manufacturing,* 26**,** 381-391.

ANGERER, S., POOLEY, R. & AYLETT, R. Self-reconfiguration of industrial mobile robots. Self-Adaptive and Self-Organizing Systems (SASO), 2010 4th IEEE International Conference on, 2010. IEEE, 64-73.

ASHINO, T. & FUJITA, M. 2006. Definition of a web ontology for design-oriented material selection. *Data Science Journal,* 5**,** 52-63.

BALENA, F. & FOREWORD BY-FAWCETTE, J. 1999. *Programming Microsoft Visual Basic 6.0*, Microsoft Press.

BANDEIRA, J., BITTENCOURT, I. I., ESPINHEIRA, P. & ISOTANI, S. 2016. FOCA: A Methodology for Ontology Evaluation. *arXiv preprint arXiv:1612.03353*.

BARNES, C. J., JARED, G. E. M. & SWIFT, K. G. 2004. Decision support for sequence generation in an assembly oriented design environment. *Robotics and Computer-Integrated Manufacturing,* 20**,** 289-300.

BEN-ARIEH, D. & KRAMER, B. 1994. Computer-aided process planning for assembly: generation of assembly operations sequence. *The international journal of production research,* 32**,** 643-656.

BENGTSSON, K. & LENNARTSON, B. 2014. Flexible specification of operation behavior using multiple projections. *IEEE Transactions on Automation Science and Engineering,* 11**,** 504-515.

BENGTSSON, K., LENNARTSON, B., LJUNGKRANTZ, O. & YUAN, C. 2013. Developing control logic using aspect-oriented programming and sequence planning. *Control Engineering Practice,* 21**,** 12-22.

BENINGTON, H. D. 1983. Production of large computer programs. *Annals of the History of Computing,* 5**,** 350-361.

BENTLEY, P. & CORNE, D. 2002. *Creative evolutionary systems*, Morgan Kaufmann.

BERGERT, M. & KIEFER, J. 2010. Mechatronic data models in production engineering. *IFAC Proceedings Volumes,* 43**,** 60-65.

BI, Z. 2011. Revisiting system paradigms from the viewpoint of manufacturing sustainability. *Sustainability,* 3**,** 1323-1340.

BIKAS, C., ARGYROU, A., PINTZOS, G., GIANNOULIS, C., SIPSAS, K., PAPAKOSTAS, N. & CHRYSSOLOURIS, G. 2016. An Automated Assembly Process Planning System. *Procedia CIRP,* 44**,** 222-227.

BJÖRKELUND, A., EDSTRÖM, L., HAAGE, M., MALEC, J., NILSSON, K., NUGUES, P., ROBERTZ, S. G., STÖRKLE, D., BLOMDELL, A. & JOHANSSON, R. On the integration of skilled robot motions for productivity in manufacturing. Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium on, 2011a. IEEE, 1-9.

BJÖRKELUND, A., MALEC, J., NILSSON, K. & NUGUES, P. 2011b. Knowledge and skill representations for robotized production. *IFAC Proceedings Volumes,* 44**,** 8999-9004.

BLESSING, L. T. & CHAKRABARTI, A. 2009. *DRM: A Design Reseach Methodology*, Springer.

BOCK, C. & GRUNINGER, M. 2005. PSL: A semantic domain for flow models. *Software & Systems Modeling,* 4**,** 209-231.

BOEHM, B. W. 1988. A spiral model of software development and enhancement. *Computer,* 21**,** 61-72.

BORGO, S. & LEITÃO, P. 2007. Foundations for a core ontology of manufacturing. *Ontologies.* Springer.

BORST, P., AKKERMANS, H. & TOP, J. 1997. Engineering ontologies. *International Journal of Human-Computer Studies,* 46**,** 365-406.

BOURJAULT, A. L., A; 1986. Modelling an assembly process. *Int Conf Autom Manuf Ind.* IEEE.

BRACHMAN, R. J., LEVESQUE, H.J. 2004. *Knowledge Representation and Reasoning*, Morgan Kaufmann.

BRANK, J., GROBELNIK, M. & MLADENIĆ, D. 2005. A survey of ontology evaluation techniques.

ÇAĞATAY BAYINDIR, K., GÖZÜKÜÇÜK, M. A. & TEKE, A. 2011. A comprehensive overview of hybrid electric vehicle: Powertrain configurations, powertrain control techniques and electronic control units. *Energy Conversion and Management,* 52**,** 1305-1313.

CAREY, P., FARRELL, J., HUI, M. & SULLIVAN, B. 2001. Heyde's MODAPTS: A language of work. *Heyde Dynamics Party ltd***,** 27-94.

CHANDRASEGARAN, S. K., RAMANI, K., SRIRAM, R. D., HORVÁTH, I., BERNARD, A., HARIK, R. F. & GAO, W. 2013. The evolution, challenges, and future of knowledge representation in product design systems. *Computer-aided design,* 45**,** 204-228.

CHATFIELD, C. S. & JOHNSON, T. D. 2010. *Microsoft Project 2010 step by step*, Pearson Education.

CHAVEZ, H. M., SHEN, W., FRANCE, R. B., MECHLING, B. A. & LI, G. 2016. An Approach to Checking Consistency between UML Class Model and Its Java Implementation. *IEEE Transactions on Software Engineering,* 42**,** 322-344.

CHEN, G., ZHOU, J., CAI, W., LAI, X., LIN, Z. & MENASSA, R. 2006. A framework for an automotive body assembly process design system. *Computer-Aided Design,* 38**,** 531-539.

CHEN, W.-C., HSU, Y.-Y., HSIEH, L.-F. & TAI, P.-H. 2010. A systematic optimization approach for assembly sequence planning using Taguchi method, DOE, and BPNN. *Expert Systems with Applications,* 37**,** 716-726.

CHHIM, P., CHINNAM, R. B. & SADAWI, N. 2017. Product design and manufacturing process based ontology for manufacturing knowledge reuse. *Journal of Intelligent Manufacturing***,** 1-12.

CHINNATHAI, M. K., GÜNTHER, T., AHMAD, M., STOCKER, C., RICHTER, L., SCHREINER, D., VERA, D., REINHART, G. & HARRISON, R. 2017. An application of physical flexibility and software reconfigurability for the automation of battery module assembly. *Procedia CIRP*.

CHOI, S. S., YOON, T. H. & NOH, S. D. 2010. XML-based neutral file and PLM integrator for PPR information exchange between heterogeneous PLM systems. *International Journal of Computer Integrated Manufacturing,* 23**,** 216-228.

CHUNGOORA, N., YOUNG, R. I., GUNENDRAN, G., PALMER, C., USMAN, Z., ANJUM, N. A., CUTTING-DECELLE, A.-F., HARDING, J. A. & CASE, K. 2013. A model-driven ontology approach for manufacturing system interoperability and knowledge sharing. *Computers in Industry,* 64**,** 392-401.

CLARK, K. B. & FUJIMOTO, T. 1991. *Product development performance: Strategy, organization, and management in the world auto industry*, Harvard Business Press.

DACONTA, M. C., SMITH, K. T. & OERST, L. J. 2004. The Semantic Web: a guide to the future of XML, Web services, and knowledge management. *Computing Reviews,* 45**,** 778-779.

DARTIGUES, C., GHODOUS, P., GRUNINGER, M., PALLEZ, D. & SRIRAM, R. 2007. CAD/CAPP integration using feature ontology. *Concurrent Engineering,* 15**,** 237-249.

DE FAZIO, T. L. & WHITNEY, D. E. 1987. Simplified generation of all mechanical assembly sequences. *Robotics and Automation, IEEE Journal of,* 3**,** 640-658.

DELAMER, I. M. & LASTRA, J. M. Ontology modeling of assembly processes and systems using semantic web services. Industrial Informatics, 2006 IEEE International Conference on, 2006. IEEE, 611-617.

DEMOLY, F., DUTARTRE, O., YAN, X.-T., EYNARD, B., KIRITSIS, D. & GOMES, S. 2013. Product relationships management enabler for concurrent engineering and product lifecycle management. *Computers in Industry,* 64**,** 833-848.

DEMOLY, F., MONTICOLO, D., EYNARD, B., RIVEST, L. & GOMES, S. 2010. Multiple viewpoint modelling framework enabling integrated product–process design. *International Journal on Interactive Design and Manufacturing (IJIDeM),* 4**,** 269-280.

DEMOLY, F., YAN, X.-T., EYNARD, B., RIVEST, L. & GOMES, S. 2011. An assembly oriented design framework for product structure engineering and assembly sequence planning. *Robotics and Computer-Integrated Manufacturing,* 27**,** 33-46.

DESHAYES, L., FOUFOU, S. & GRUNINGER, M. 2007. An ontology architecture for standards integration and conformance in manufacturing. *Advances in Integrated Design and Manufacturing in Mechanical Engineering II***,** 261-276.

DIETRICH, U., SCHULZ, T. & YARAMANOGLU, N. 2002. Bringing real and virtual worlds together in the manufacturing process. *Journal of Advanced Manufacturing Systems,* 1**,** 51-65.

DIN 2003. DIN 8580 Manufacturing processes - Terms and definitions.

DO, N. 2015. Integration of engineering change objects in product data management databases to support engineering change analysis. *Computers in Industry,* 73**,** 69-81.

DONG, T., TONG, R., ZHANG, L. & DONG, J. 2007. A knowledge-based approach to assembly sequence planning. *The International Journal of Advanced Manufacturing Technology,* 32**,** 1232-1244.

DRATH, R., LÜDER, A., PESCHKE, J. & HUNDT, L. AutomationML-the glue for seamless automation engineering. Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on, 2008. IEEE, 616-623.

E. CARTER, D. & S. BAKER, B. 1992. *Concurrent Engineering: Product Development Environment for the 1990s*, Addison-Wesley Publishing Company.

ECKERT, C., WYNN, D. & CLARKSON, J. 2009. Design customisation in multi-project environments: Using process simulation to explore the issues.

EL KADIRI, S. & KIRITSIS, D. 2015. Ontologies in the context of product lifecycle management: state of the art literature review. *International Journal of Production Research,* 53**,** 5657-5668.

ELMARAGHY, H., SCHUH, G., ELMARAGHY, W., PILLER, F., SCHÖNSLEBEN, P., TSENG, M. & BERNARD, A. 2013. Product variety management. *CIRP Annals-Manufacturing Technology,* 62**,** 629-652.

ELMARAGHY, H. & WIENDAHL, H.-P. 2014. Changeable Manufacturing. *CIRP Encyclopedia of Production Engineering.* Springer.

ELMARAGHY, H. A. 2006. Flexible and reconfigurable manufacturing systems paradigms. *International Journal of Flexible Manufacturing Systems,* 17**,** 261-276.

ELMARAGHY, H. A. 2012. *Enabling manufacturing competitiveness and economic sustainability*, Springer.

ELMARAGHY, W., ELMARAGHY, H., TOMIYAMA, T. & MONOSTORI, L. 2012. Complexity in engineering design and manufacturing. *CIRP Annals-Manufacturing Technology,* 61**,** 793-814.

ENSAN, F. & DU, W. 2011. A knowledge encapsulation approach to ontology modularization. *Knowledge and information systems,* 26**,** 249-283.

ESTEFAN, J. A. 2007. Survey of model-based systems engineering (MBSE) methodologies. *Incose MBSE Focus Group,* 25.

EXEL, L., FREY, G., WOLF, G. & OPPELT, M. Re-use of existing simulation models for DCS engineering via the Functional Mock-up Interface. Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), 2014. IEEE, 1-4.

FELDMANN, S., HERZIG, S. J., KERNSCHMIDT, K., WOLFENSTETTER, T., KAMMERL, D., QAMAR, A., LINDEMANN, U., KRCMAR, H., PAREDIS, C. J. & VOGEL-HEUSER, B. 2015. Towards effective management of inconsistencies in model-based engineering of automated production systems. *IFAC-PapersOnLine,* 48**,** 916-923.

FENG, S. C. & SONG, E. Y. 2003. A manufacturing process information model for design and process planning integration. *Journal of Manufacturing Systems,* 22**,** 1.

FENVES, S. J., FOUFOU, S., BOCK, C. & SRIRAM, R. D. 2008. CPM2: a core model for product data. *Journal of Computing and Information Science in Engineering,* 8**,** 014501.

FERREIRA, P. & LOHSE, N. Configuration model for evolvable assembly systems. 4th CIRP Conference On Assembly Technologies And Systems, 2012.

FERREIRA, P., LOHSE, N., RAZGON, M., LARIZZA, P. & TRIGGIANI, G. Skill based configuration methodology for evolvable mechatronic systems. IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society, 2012. IEEE, 4366-4371.

FINKELSTEIN, A. C., GABBAY, D., HUNTER, A., KRAMER, J. & NUSEIBEH, B. 1994. Inconsistency handling in multiperspective specifications. *IEEE Transactions on Software Engineering,* 20**,** 569-578.

FORSBERG, K. & MOOZ, H. The relationship of system engineering to the project cycle. INCOSE International Symposium, 1991. Wiley Online Library, 57-65.

FRICKE, E., GEBHARD, B., NEGELE, H. & IGENBERGS, E. 2000. Coping with changes: causes, findings, and strategies. *Systems Engineering,* 3**,** 169-179.

GAUSEMEIER, J., SCHÄFER, W., GREENYER, J., KAHL, S., POOK, S. & RIEKE, J. Management of cross-domain model consistency during the development of advanced mechatronic systems. DS 58-6: Proceedings of ICED 09, the 17th International Conference on Engineering Design, Vol. 6, Design Methods and Tools (pt. 2), Palo Alto, CA, USA, 24.-27.08. 2009, 2009.

GÓMEZ-PÉREZ, A. 2001. Evaluation of ontologies. *International Journal of intelligent systems,* 16**,** 391-409.

GRUBER, T. R. 1993. A translation approach to portable ontology specifications. *Knowledge acquisition,* 5**,** 199-220.

GRÜNINGER, M. 2004. Ontology of the process specification language. *Handbook on ontologies.* Springer.

GRÜNINGER, M. & KOPENA, J. B. Planning and the process specification language. Proceedings of the ICAPS 2005 Workshop on the Role of Ontologies in Planning and Scheduling, 2005. 22-29.

GUARINO, N. Formal ontology and information systems. Proceedings of FOIS, 1998. 81-97.

GUARINO, N., OBERLE, D. & STAAB, S. 2009. What is an Ontology? *Handbook on ontologies.* Springer.

HAAGE, M., MALEC, J., NILSSON, A., NILSSON, K. & NOWACZYK, S. Declarative-knowledge-based reconfiguration of automation systems using a blackboard architecture. Eleventh Scandinavian Conference on Artificial Intelligence, 2011. IOS Press, 163-172.

HAMRAZ, B., CALDWELL, N. H. & CLARKSON, P. J. 2013. A holistic categorization framework for literature on engineering change management. *Systems Engineering,* 16**,** 473-505.

HANKEL, M. & REXROTH, B. 2015. Industrie 4.0: The Reference Architectural Model Industrie 4.0 (RAMI 4.0).

HANSSEN, D. H. 2015. IEC 61131-3. *Programmable Logic Controllers: A Practical Approach to IEC 61131-3 using CODESYS***,** 152-186.

HARRISON, R., VERA, D. & AHMAD, B. Engineering Methods and Tools for Cyber–Physical Automation Systems.

HARRISON, R., VERA, D. & AHMAD, B. 2016. Engineering methods and tools for cyber–physical automation systems. *Proceedings of the IEEE,* 104**,** 973-985.

HARRISON;, S. K. M. A. K. A. R. 2017. The Cyber-Physical e-machine Manufacturing System: Virtual Engineering for Complete Lifecycle Support. *Procedia CIRP*.

HART, L. Introduction to Model-Based System Engineering (MBSE) and SysML. Delaware Valley INCOSE Chapter Meeting, 2015 Delaware.

HASAN, B. & WIKANDER, J. 2016. Product Feature Modelling for Integrating Product Design and Assembly Process Planning. *World Academy of Science, Engineering and Technology, International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering,* 10**,** 1693-1703.

HASAN, B. & WIKANDER, J. Features Extraction from CAD as a Basis for Assembly Process Planning. Doctoral Conference on Computing, Electrical and Industrial Systems, 2017. Springer, 144-153.

HASAN, B., WIKANDER, J. & ONORI, M. Utilizing Assembly Features for determination of Grasping Skill in Assembly System. Mechatronics Conference 2014 in Karlstad, June 16-18, 2014. Curran Associates, Inc, 399-406.

HASAN, B., WIKANDER, J. & ONORI, M. 2016a. Assembly design semantic recognition using SolidWorks-API. *Int. J. Mech. Eng. Robot. Res,* 5**,** 280-287.

HASAN, B., WIKANDER, J. & ONORI, M. Ontological approach to share product design semantics for an assembly. 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2016, Porto, Portugal, 9 November 2016 through 11 November 2016, 2016b. SciTePress, 104-111.

HAYES, J. 2014. *The theory and practice of change management*, Palgrave Macmillan.

HEFLIN, J. & HENDLER, J. Dynamic ontologies on the web. AAAI/IAAI, 2000. 443-449.

HEHENBERGER, P., EGYED, A. & ZEMAN, K. Consistency checking of mechatronic design models. ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 2010. American Society of Mechanical Engineers, 1141-1148.

HERZIG, S. J. & PAREDIS, C. J. 2014. A conceptual basis for inconsistency management in Model-Based Systems Engineering. *Procedia CIRP,* 21**,** 52-57.

HEWETT, A. 2009. Product Lifecycle Management (PLM): Critical Issues and Challenges in Implementation. *In:* NAMBISAN, S. (ed.) *Information Technology and Product Development.* Boston, MA: Springer US.

HLOMANI, H. & STACEY, D. 2014. Approaches, methods, metrics, measures, and subjectivity in ontology evaluation: A survey. *Semantic Web Journal***,** 1-5.

HOMEM DE MELLO, L. S. & SANDERSON, A. C. 1991. A correct and complete algorithm for the generation of mechanical assembly sequences. *Robotics and Automation, IEEE Transactions on,* 7**,** 228-240.

HORROCKS, I., PATEL-SCHNEIDER, P. F., BOLEY, H., TABET, S., GROSOF, B. & DEAN, M. 2004. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission,* 21**,** 79.

HSU, Y.-Y., TAI, P.-H., WANG, M.-W. & CHEN, W.-C. 2011. A knowledge-based engineering system for assembly sequence planning. *The International Journal of Advanced Manufacturing Technology,* 55**,** 763-782.

HU, S. J., KO, J., WEYAND, L., ELMARAGHY, H., LIEN, T., KOREN, Y., BLEY, H., CHRYSSOLOURIS, G., NASR, N. & SHPITALNI, M. 2011. Assembly system design and operations for product variety. *CIRP Annals-Manufacturing Technology,* 60**,** 715-733.

HUANG, G. & MAK, K. 1999. Current practices of engineering change management in UK manufacturing industries. *International Journal of Operations & Production Management,* 19**,** 21-37.

HUANG, G., YEE, W. & MAK, K. 2001. Development of a web-based system for engineering change management. *Robotics and Computer-Integrated Manufacturing,* 17**,** 255-267.

HUANG, G. Q., YEE, W. Y. & MAK, K. L. 2003. Current practice of engineering change management in Hong Kong manufacturing industries. *Journal of Materials Processing Technology,* 139**,** 481-487.

HUCKABY, J. & CHRISTENSEN, H. I. A taxonomic framework for task modeling and knowledge transfer in manufacturing robotics. Workshops at 26th AAAI conference on artificial intelligence, 2012.

IAROVYI, S., MOHAMMED, W. M., LOBOV, A., FERRER, B. R. & LASTRA, J. L. M. 2016. Cyber Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems. *Proceedings of the IEEE,* PP**,** 1-13.

IIVARI, J. 1987. A hierarchical spiral model for the software process. *ACM SIGSOFT Software Engineering Notes,* 12**,** 35-37.

ISO 2011. ISO 10303-203:2011 Industrial automation systems and integration -- Product data representation and exchange -- Part 203: Application protocol: Configuration controlled 3D design of mechanical parts and assemblies. www.iso.org.

JAISWAL, D., DEY, S., DASGUPTA, R. & MUKHERJEE, A. Spatial query handling in semantic web application: an experience report. 2015 Applications and Innovations in Mobile Computing (AIMoC), 12-14 Feb. 2015 2015. 170-175.

JARRATT, T., ECKERT, C. M., CALDWELL, N. & CLARKSON, P. J. 2011. Engineering change: an overview and perspective on the literature. *Research in engineering design,* 22**,** 103-124.

JÄRVENPÄÄ, E. 2012. *Capability-based Adaption of Production Systems in a Changing Environment.* PhD Thesis, Tampere University of Technology.

JÄRVENPÄÄ, E., LANZ, M., MELA, J. & TUOKKO, R. Studying the information sources and flows in a company-Support for the development of new intelligent systems. Proceedings of the 20th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM 2010, July 12-14, 2010, California State University East Bay, USA, 2010.

JÄRVENPÄÄ, E., SILTALA, N. & LANZ, M. Formal resource and capability descriptions supporting rapid reconfiguration of assembly systems. 2016 IEEE International Symposium on Assembly and Manufacturing (ISAM), 21-22 Aug. 2016 2016. 120-125.

JAVED, M., ABGAZ, Y. M. & PAHL, C. 2013. Ontology Change Management and Identification of Change Patterns. *Journal on Data Semantics,* 2**,** 119-143.

JUN, Y., LIU, J., NING, R. & ZHANG, Y. 2005. Assembly process modeling for virtual assembly process planning. *International Journal of Computer Integrated Manufacturing,* 18**,** 442-451.

KALIBATIENE, D. & VASILECAS, O. Survey on ontology languages. International Conference on Business Informatics Research, 2011. Springer, 124-141.

KASHKOUSH, M. & ELMARAGHY, H. 2014. Consensus tree method for generating master assembly sequence. *Production Engineering,* 8**,** 233-242.

KASHKOUSH, M. & ELMARAGHY, H. 2015. Knowledge-based model for constructing master assembly sequence. *Journal of Manufacturing Systems,* 34**,** 43-52.

KEDDIS, N., KAINZ, G., BUCKL, C. & KNOLL, A. Towards adaptable manufacturing systems. Industrial Technology (ICIT), 2013 IEEE International Conference on, 2013. IEEE, 1410-1415.

KEDDIS, N., KAINZ, G. & ZOITL, A. Capability-based planning and scheduling for adaptable manufacturing systems. Emerging Technology and Factory Automation (ETFA), 2014 IEEE, 2014. IEEE, 1-8.

KERNSCHMIDT, K. & VOGEL-HEUSER, B. An interdisciplinary SysML based modeling approach for analyzing change influences in production plants to support the engineering. Automation Science and Engineering (CASE), 2013 IEEE International Conference on, 17-20 Aug. 2013 2013. 1113-1118.

KICINGER, R., ARCISZEWSKI, T. & DE JONG, K. 2005. Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers & Structures,* 83**,** 1943-1978.

KIM, K.-Y., MANLEY, D. G. & YANG, H. 2006. Ontology-based assembly design and information sharing for collaborative product development. *Computer-Aided Design,* 38**,** 1233-1250.

KNUTILLA, A., SCHLENOFF, C., RAY, S., POLYAK, S. T., TATE, A., CHEAH, S. C. & ANDERSON, R. C. 1998. Process specification language: An analysis of existing representations. *National Institute of Standards and Technology (NIST), Gaithersburg (MD), NISTIT,* 6160.

KOCH, J., GRITSCH, A. & REINHART, G. 2016. Process design for the management of changes in manufacturing: Toward a Manufacturing Change Management process. *CIRP Journal of Manufacturing Science and Technology,* 14**,** 10-19.

KOREN, Y., HEISEL, U., JOVANE, F., MORIWAKI, T., PRITSCHOW, G., ULSOY, G. & VAN BRUSSEL, H. 1999. Reconfigurable manufacturing systems. *CIRP Annals-Manufacturing Technology,* 48**,** 527-540.

KOREN, Y. & SHPITALNI, M. 2010. Design of reconfigurable manufacturing systems. *Journal of manufacturing systems,* 29**,** 130-141.

KOVALENKO, O., WIMMER, M., SABOU, M., LUDER, A., EKAPUTRA, F. J. & BIFFL, S. Modeling AutomationML: Semantic Web Technologies vs. Model-Driven Engineering. Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on, 2015. IEEE, 1-4.

KRIMA, S., BARBAU, R., FIORENTINI, X., RACHURI, S., FOUFOU, S. & SRIRAM, R. D. 2009. OntoSTEP: OWL-DL ontology for STEP.

KUEHN, W. 2006. Digital factory: integration of simulation enhancing the product and production process towards operative control and optimisation. *International Journal of Simulation,* 7**,** 27-39.

KUHN, W. Digital factory-simulation enhancing the product and production engineering process. Simulation Conference, 2006. WSC 06. Proceedings of the Winter, 2006. IEEE, 1899-1906.

LABROUSSE, M. & BERNARD, A. 2008. FBS-PPRE, an enterprise knowledge lifecycle model. *Methods and tools for effective knowledge life-cycle-management.* Springer.

LANZ, M. 2010. Logical and semantic foundations of knowledge representation for assembly and manufacturing processes. *Tampereen teknillinen yliopisto. Julkaisu-Tampere University of Technology. Publication; 903*.

LASTRA, J. 2004. Reference Mechatronic Architecture for Actor-Based Assembly Systems, PhD Thesis. *Tampere University of Technology, Tampere*.

LEE, C., LEEM, C. S. & HWANG, I. 2011a. PDM and ERP integration methodology using digital manufacturing to support global manufacturing. *The International Journal of Advanced Manufacturing Technology,* 53**,** 399-409.

LEE, J. Y., KIM, G. Y. & NOH, S. D. 2011b. Integration Framework and PPR+H Hub for DiFac. *In:* CANETTA, L., REDAELLI, C. & FLORES, M. (eds.) *Digital Factory for Human-oriented Production Systems: The Integration of International Research Projects.* London: Springer London.

LEE, S., HARRISON, R., WEST, A. & ONG, M. 2007. A component-based approach to the design and implementation of assembly automation system. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture,* 221**,** 763-773.

LEITÃO, P. 2009. Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence,* 22**,** 979-991.

LEITÃO, P., BARBOSA, J., PEREIRA, A., BARATA, J. & COLOMBO, A. W. Specification of the PERFoRM architecture for the seamless production system reconfiguration. IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, 23-26 Oct. 2016 2016. 5729-5734.

LEITAO, P. J. P. 2004. *An agile and adaptive holonic architecture for manufacturing control.* University of Porto.

LEMAIGNAN, S., SIADAT, A., DANTAN, J.-Y. & SEMENENKO, A. MASON: A proposal for an ontology of manufacturing domain. IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06), 2006. IEEE, 195-200.

LENNARTSON, B., BENGTSSON, K., YUAN, C., ANDERSSON, K., FABIAN, M., FALKMAN, P. & AKESSON, K. 2010. Sequence planning for integrated product, process and automation design. *IEEE Transactions on Automation Science and Engineering,* 7**,** 791-802.

LIN, H.-K., HARDING*, J. A. & SHAHBAZ, M. 2004. Manufacturing system engineering ontology for semantic interoperability across extended project teams. *International journal of production research,* 42**,** 5099-5118.

LIU, X. Identification and check of inconsistencies between UML diagrams. Computer Sciences and Applications (CSA), 2013 International Conference on, 2013. IEEE, 487-490.

LOHSE, N. 2006. *Towards an ontology framework for the integrated design of modular assembly systems.* University of Nottingham.

LOHSE, N., HIRANI, H. & RATCHEV, S. 2005. Equipment ontology for modular reconfigurable assembly systems. *International journal of flexible manufacturing systems,* 17**,** 301-314.

LOHSE, N., RATCHEV, S. & VALTCHANOV, G. 2004. Towards web-enabled design of modular assembly systems. *Assembly Automation,* 24**,** 270-279.

MAKRIS, S., MICHALOS, G. & CHRYSSOLOURIS, G. 2012. Virtual commissioning of an assembly cell with cooperating robots. *Advances in Decision Sciences,* 2012.

MALEC, J., NILSSON, A., NILSSON, K. & NOWACZYK, S. Knowledge-based reconfiguration of automation systems. Automation Science and Engineering, 2007. CASE 2007. IEEE International Conference on, 2007. IEEE, 170-175.

MARTINEZ-CRUZ, C., BLANCO, I. J. & VILA, M. A. 2012. Ontologies versus relational databases: are they so different? A comparison. *Artificial Intelligence Review,* 38**,** 271-290.

MASOLO, C., BORGO, S., GANGEMINI, A., GUARINO, N., OLTRAMARI, A. & SCHNEIDER, L. 2003. The wonderweb library of fundational ontologies and the dolce ontology. wonderweb deliverable d18, final report (vr. 1.0. 31-12-2003). *The WonderWeb Library of Fundational Ontologies and the DOLCE ontology. WonderWeb Deliverable D18, Final Report (vr. 1. 0. 31-12-2003)*.

MATHEW, A. T. & RAO, C. 2010. A novel method of using API to generate liaison relationships from an assembly. *Journal of Software Engineering and Applications,* 3**,** 167.

MATSOKIS, A. & KIRITSIS, D. 2010. An ontology-based approach for Product Lifecycle Management. *Computers in Industry,* 61**,** 787-797.

MATSUDA, M. & WANG, Q. 2010. Software interoperability tools: Standardized capability-profiling methodology ISO16100. *Enterprise architecture, integration and interoperability***,** 140-151.

MAULL, R., HUGHES, D. & BENNETT, J. 1992. The role of the bill-of-materials as a CAD/CAPM interface and the key importance of engineering change control. *Computing & Control Engineering Journal,* 3**,** 63-70.

MAYER, R. J. 1992. IDEF0 function modeling. *A Reconstruction of the Original Air Force Wright Aeronautical Laboratory Technical Report, AFWAL-TR-81-4023 (The IDEF0 Yellow Book), Knowledge-Based System Inc, College Station, TX*.

MCGUINNESS, D. L. & VAN HARMELEN, F. 2004. OWL web ontology language overview. *W3C recommendation,* 10**,** 2004.

MCMAHON, C. A. 1994. Observations on modes of incremental change in design. *Journal of Engeering Design,* 5**,** 195-209.

MEHTA, V. & COOPER, J. S. 2003. Review and analysis of PEM fuel cell design and manufacturing. *Journal of Power Sources,* 114**,** 32-53.

MENS, T. 2002. A state-of-the-art survey on software merging. *IEEE transactions on software engineering,* 28**,** 449-462.

MICHNIEWICZ, J. & REINHART, G. 2014. Cyber-physical Robotics–Automated Analysis, Programming and Configuration of Robot Cells based on Cyber-physical-systems. *Procedia Technology,* 15**,** 566-575.

MICHNIEWICZ, J. & REINHART, G. 2015. Cyber-Physical-Robotics–Modelling of modular robot cells for automated planning and execution of assembly tasks. *Mechatronics*.

MICHNIEWICZ, J., REINHART, G. & BOSCHERT, S. 2016. CAD-Based Automated Assembly Planning for Variable Products in Modular Production Systems. *Procedia CIRP,* 44**,** 44-49.

MILICIC, A., PERDIKAKIS, A., KADIRI, S. E. & KIRITSIS, D. 2013. PLM Ontology Exploitation through Inference and Statistical Analysis A Case Study for LCC. *IFAC Proceedings Volumes,* 46**,** 1004-1008.

MILLER, G. A. 1995. WordNet: a lexical database for English. *Communications of the ACM,* 38**,** 39-41.

MIN, B.-K., HUANG, Z., PASEK, Z. J., YIP-HOI, D., HUSTED, F. & MARKER, S. 2002. Integration of real-time control simulation to a virtual manufacturing environment. *Journal of advanced manufacturing systems,* 1**,** 67-87.

MONOSTORI, L., KÁDÁR, B., BAUERNHANSL, T., KONDOH, S., KUMARA, S., REINHART, G., SAUER, O., SCHUH, G., SIHN, W. & UEDA, K. 2016. Cyber-physical systems in manufacturing. *CIRP Annals - Manufacturing Technology,* 65**,** 621-641.

MOURTZIS, D. & DOUKAS, M. 2014. The Evolution of Manufacturing Systems: From Craftsmanship to the. *Handbook of Research on Design and Management of Lean Production Systems***,** 1.

OBRST, L., WERNER, C., INDERJEET, M., STEVE, R. & SMITH, B. 2007. The Evaluation of Ontologies: Toward Improved Semantic Interoperability. Dans JO Christopher, Baker, & K.-H. Cheung. *Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences*.

OLIVEIRA, J. A. B. D. 2003. Coalition based approach for shop floor agility–a multiagent approach.

ONORI, M., LOHSE, N., BARATA, J. & HANISCH, C. 2012. The IDEAS project: plug & produce at shop-floor level. *Assembly automation,* 32**,** 124-134.

OTTO, K. N. & WOOD, K. L. 1998. Product evolution: a reverse engineering and redesign methodology. *Research in Engineering Design,* 10**,** 226-243.

PABADIS'PROMISE, F. 2006. D3. 1 Development of manufacturing ontology, project deliverable. *The PABADIS'PROMISE consortium*.

PAHL, G. & BEITZ, W. 2013. *Engineering design: a systematic approach*, Springer Science & Business Media.

PANETTO, H., DASSISTI, M. & TURSI, A. 2012. ONTO-PDM: product-driven ONTOlogy for Product Data Management interoperability within manufacturing process environment. *Advanced Engineering Informatics,* 26**,** 334-348.

PFROMMER, J., SCHLEIPEN, M. & BEYERER, J. PPRS: Production skills and their relation to product, process, and resource. Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on, 2013. IEEE, 1-4.

PFROMMER, J., STOGL, D., ALEKSANDROV, K., SCHUBERT, V. & HEIN, B. Modelling and orchestration of service-based manufacturing systems via skills. Emerging Technology and Factory Automation (ETFA), 2014 IEEE, 2014. IEEE, 1-4.

PINTZOS, G., TRIANTAFYLLOU, C., PAPAKOSTAS, N., MOURTZIS, D. & CHRYSSOLOURIS, G. 2016. Assembly precedence diagram generation through assembly tiers determination. *International Journal of Computer Integrated Manufacturing,* 29**,** 1045-1057.

PROCTOR, F. M., VAN DER HOORN, G. & LIPMAN, R. 2016. Automating Robot Planning Using Product and Manufacturing Information. *Procedia CIRP,* 43**,** 208-213.

PRUD'HOMMEAUX, E. & SEABORNE, A. 2008. SPARQL query language for RDF. *W3C recommendation,* 15.

PUTTONEN, J., LOBOV, A. & LASTRA, J. L. M. Maintaining a Dynamic View of Semantic Web Services Representing Factory Automation Systems. Web Services (ICWS), 2013 IEEE 20th International Conference on, June 28 2013-July 3 2013 2013. 419-426.

PYZDEK, T. & KELLER, P. A. 2014. *The six sigma handbook*, McGraw-Hill Education New York.

QUINTANA, V., RIVEST, L., PELLERIN, R. & KHEDDOUCI, F. 2012. Re-engineering the Engineering Change Management process for a drawing-less environment. *Computers in Industry,* 63**,** 79-90.

RACHURI, S., HAN, Y.-H., FOUFOU, S., FENG, S. C., ROY, U., WANG, F., SRIRAM, R. D. & LYONS, K. W. 2006. A model for capturing product assembly information. *Journal of Computing and Information Science in Engineering,* 6**,** 11-21.

RAMIS FERRER, B., AHMAD, B., LOBOV, A., VERA, D., MARTINEZ LASTRA, J. L. & HARRISON, R. 2015a. An approach for knowledge-driven product, process and resource mappings for assembly automation. *2015 IEEE International Conference on Automation Science and Engineering (CASE).*

RAMIS FERRER, B., AHMAD, B., LOBOV, A., VERA, D., MARTINEZ LASTRA, J. L. & HARRISON, R. 2015b. A knowledge-based solution for automatic mapping in component based automation systems. *13th IEEE International Conference on Industrial Informatics (INDIN).*

RAMIS FERRER, B., AHMAD, B., VERA, D., LOBOV, A., HARRISON, R. & MARTÍNEZ LASTRA, J. L. 2016. Product, process and resource model coupling for knowledge-driven assembly automation. *at-Automatisierungstechnik,* 64**,** 231-243.

RAMPERSAD, H. K. 1994. *Integrated and Simultaneous Design for Robotic Assembly: Product Development, Planning*, John Wiley & Sons, Inc.

RASHID, M. F. F., HUTABARAT, W. & TIWARI, A. 2012. A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. *The International Journal of Advanced Manufacturing Technology,* 59**,** 335-349.

RAZA, M. B. & HARRISON, R. Ontological knowledge based system for product, process and resource relationships in automotive industry. Proceedings of the 1st International Workshop on Ontology and Semantic Web for Manufacturing, 2011. 23-36.

ROYCE, W. W. Managing the development of large software systems: concepts and techniques. Proceedings of the 9th international conference on Software Engineering, 1987. IEEE Computer Society Press, 328-338.

RUPARELIA, N. B. 2010. Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes,* 35**,** 8-13.

SACCO, M., PEDRAZZOLI, P. & TERKAJ, W. VFF: virtual factory framework. Proceedings of 16th International Conference on Concurrent Enterprising, Lugano, Switzerland, 2010. 21-23.

SANDERSON, A. C., DE MELLO, L. S. H. & ZHANG, H. 1990. Assembly sequence planning. *AI Magazine,* 11**,** 62.

SCHLEIPEN, M., PFROMMER, J., ALEKSANDROV, K., STOGL, D., ESCAIDA, S., BEYERER, J. & HEIN, B. Automationml to describe skills of production plants based on the ppr concept. 3rd AutomationML user conference, 2014.

SEABORNE, A., MANJUNATH, G., BIZER, C., BRESLIN, J., DAS, S., DAVIS, I., HARRIS, S., IDEHEN, K., CORBY, O. & KJERNSMO, K. 2008. SPARQL/Update: A language for updating RDF graphs. *W3c member submission,* 15.

SHANKAR, P., MORKOS, B. & SUMMERS, J. D. 2012. Reasons for change propagation: a case study in an automotive OEM. *Research in Engineering Design,* 23**,** 291-303.

SHARAF, O. Z. & ORHAN, M. F. 2014. An overview of fuel cell technology: Fundamentals and applications. *Renewable and Sustainable Energy Reviews,* 32**,** 810-853.

SIRIN, E., PARSIA, B., GRAU, B. C., KALYANPUR, A. & KATZ, Y. 2007. Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web,* 5**,** 51-53.

SMITH, B. & GRENON, P. 2002. Basic formal ontology. *Draft. Downloadable at http://ontology. buffalo. edu/bfo*.

SPANOUDAKIS, G. & ZISMAN, A. 2001. Inconsistency management in software engineering: Survey and open research issues. *Handbook of software engineering and knowledge engineering,* 1**,** 329-380.

STAAB, S. & STUDER, R. 2010. *Handbook on ontologies*, Springer Science & Business Media.

STAMATIS, D. H. 2002. *Six sigma and beyond: design for six sigma*, CRC Press.

STARK, R., KIND, S. & NEUMEYER, S. 2017. Innovations in digital modelling for next generation manufacturing system design. *CIRP Annals-Manufacturing Technology*.

STENMARK, M. & MALEC, J. 2015. Knowledge-based instruction of manipulation tasks for industrial robotics. *Robotics and Computer-Integrated Manufacturing,* 33**,** 56-67.

SU, Q. & SMITH, S. 2003. An integrated framework for assembly-oriented product design and optimization. *Journal of Industrial Technology,* 19.

SUH, S.-H., SEO, Y., LEE, S.-M., CHOI, T.-H., JEONG, G.-S. & KIM, D.-Y. 2003. Modelling and Implementation of Internet-Based Virtual Machine Tools. *The International Journal of Advanced Manufacturing Technology,* 21**,** 516-522.

TECHNOLOGY, N. I. F. S. A. 2005. CPM: A core model for product data.

TERKAJ, W., TOLIO, T. & URGO, M. 2015. A virtual factory approach for in situ simulation to support production and maintenance planning. *CIRP Annals-Manufacturing Technology,* 64**,** 451-454.

TOLIO, T., CEGLAREK, D., ELMARAGHY, H., FISCHER, A., HU, S., LAPERRIÈRE, L., NEWMAN, S. T. & VÁNCZA, J. 2010. SPECIES—Co-evolution of products, processes and production systems. *CIRP Annals-Manufacturing Technology,* 59**,** 672-693.

TOLIO, T., SACCO, M., TERKAJ, W. & URGO, M. 2013. Virtual Factory: An Integrated Framework for Manufacturing Systems Design and Analysis. *Procedia CIRP,* 7**,** 25-30.

USMAN, Z. 2012. *A manufacturing core concepts ontology to support knowledge sharing.* Loughborough University.

USMAN, Z., YOUNG, R. I. M., CHUNGOORA, N., PALMER, C., CASE, K. & HARDING, J. A manufacturing core concepts ontology for product lifecycle interoperability. International IFIP Working Conference on Enterprise Interoperability, 2011. Springer, 5-18.

USMAN, Z., YOUNG, R. I. M., CHUNGOORA, N., PALMER, C., CASE, K. & HARDING, J. A. 2013. Towards a formal manufacturing reference ontology. *International Journal of Production Research,* 51**,** 6553-6572.

VDI 1990. VDI 2860 - Assembly and handling; handling functions, handling units; terminology, definitions and symbols.

VOGEL-HEUSER, B., SCHÜTZ, D., FRANK, T. & LEGAT, C. 2014. Model-driven engineering of Manufacturing Automation Software Projects–A SysML-based approach. *Mechatronics,* 24**,** 883-897.

VRANDEČIĆ, D. 2009. Ontology evaluation. *Handbook on Ontologies.* Springer.

VYATKIN, V. 2009. The IEC 61499 standard and its semantics. *IEEE Industrial Electronics Magazine,* 3.

WANG, D., ZAMEL, N., JIAO, K., ZHOU, Y., YU, S., DU, Q. & YIN, Y. 2013. Life cycle analysis of internal combustion engine, electric and fuel cell vehicles for China. *Energy,* 59**,** 402-412.

WANG, L., ADAMSON, G., HOLM, M. & MOORE, P. 2012. A review of function blocks for process planning and control of manufacturing equipment. *Journal of manufacturing systems,* 31**,** 269-279.

WANG, L., KESHAVARZMANESH, S. & FENG, H.-Y. 2008. Design of adaptive function blocks for dynamic assembly planning and control. *Journal of Manufacturing Systems,* 27**,** 45-51.

WANG, L., KESHAVARZMANESH, S., FENG, H.-Y. & BUCHAL, R. O. 2009. Assembly process planning and its future in collaborative manufacturing: a review. *The International Journal of Advanced Manufacturing Technology,* 41**,** 132-144.

WASMER, A., STAUB, G. & VROOM, R. W. 2011. An industry approach to shared, cross-organisational engineering change handling - The road towards standards for product data processing. *Computer-Aided Design,* 43**,** 533-545.

WESER, M. & ZHANG, J. Autonomous planning for mobile manipulation services based on multi-level robot skills. 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 10-15 Oct. 2009 2009. 1999-2004.

WESTKÄMPER, E. & JENDOUBI, L. Smart factories–Manufacturing environments and systems of the future. Proceedings of the 36th CIRP International Seminar on Manufacturing Systems, 2003. 13-16.

WINKLER, D., EKAPUTRA, F. & BIFFL, S. AutomationML review support in multi-disciplinary engineering environments. 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), 6-9 Sept. 2016 2016. 1-9.

WU, W.-H., FANG, L.-C., LIN, T.-H., YEH, S.-C. & HO, C.-F. 2012. A novel CMII-based engineering change management framework: an example in Taiwan's motorcycle industry. *IEEE Transactions on Engineering Management,* 59**,** 494-505.

WYNN, D. C., CALDWELL, N. H. & CLARKSON, P. J. 2010. Can change prediction help prioritise redesign work in future engineering systems?

XU, J., LIANG, B., WANG, J., XU, X. & ZHANG, B. An approach to automatic assembly sequences generation. Proceedings of 2nd Asian Conference on Robotics and its Application, 1994. 612-615.

YANG, B., QIAO, L., ZHU, Z. & WULAN, M. 2016. A Metamodel for the Manufacturing Process Information Modeling. *Procedia CIRP,* 56**,** 332-337.

ZABLITH, F. 2008. Dynamic ontology evolution.

ZHA, X., LIM, S. & FOK, S. 1999. Development of expert system for concurrent product design and planning for assembly. *The International Journal of Advanced Manufacturing Technology,* 15**,** 153-162.

ZHA, X. F., DU, H. J. & QIU, J. H. 2001a. Knowledge-based approach and system for assembly-oriented design, Part II: the system implementation. *Engineering Applications of Artificial Intelligence,* 14**,** 239-254.

ZHA, X. F., DU, H. J. & QIU, J. H. 2001b. Knowledge-based approach and system for assembly oriented design, Part I: the approach. *Engineering Applications of Artificial Intelligence,* 14**,** 61-75.